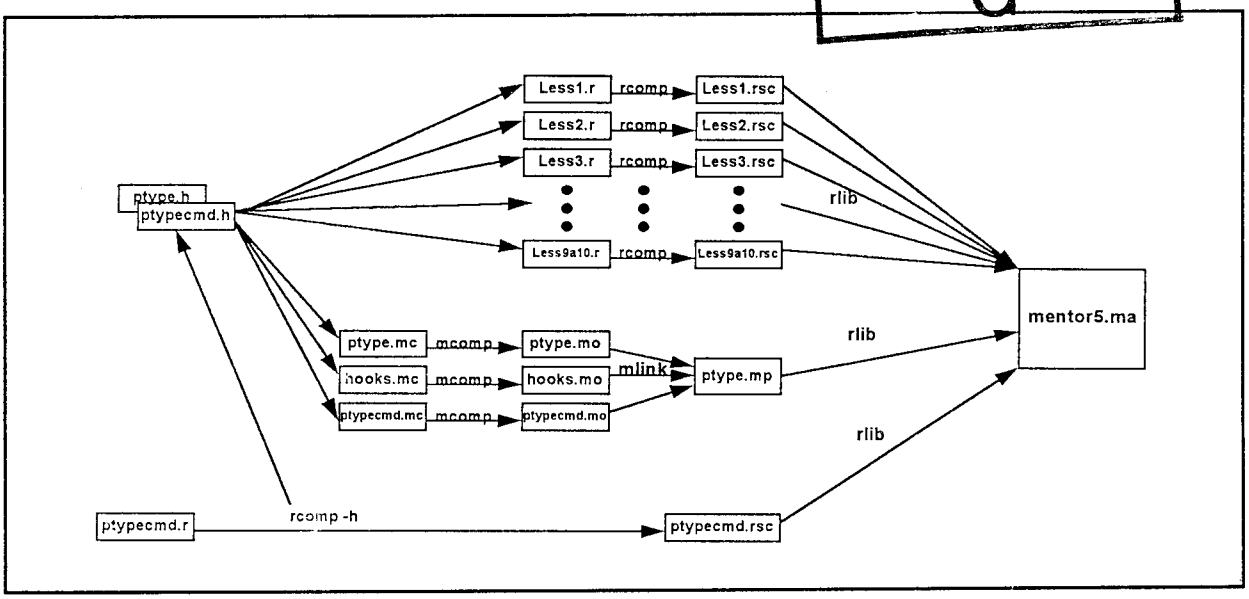
US Army Corps
of Engineers
Construction Engineering
Research Laboratories

USACERL Technical Report FF-95/06
January 1995

Effectiveness of an Electronic Performance Support System for Use With Computer-Aided Design Software

by
Doris S. Shaw
Thomas E. Mack
Daniel J. Nix
Julie L. Webster
Donald K. Hicks



Although Bentley Systems, Inc. (BSI) CAD, a computer-aided design program used in many U.S. Army Corps of Engineers offices, can produce the detailed documents necessary for planning, review, construction, and management of a facility, such software requires extensive training. Allocating the time and resources to learn such computer-based technology is a pressing productivity concern in both government and commercial settings. The U.S. Army Construction Engineering Research Laboratories (USACERL) has studied the use of Computer-Aided Instruction (CAI) to train architects and engineers to use many types of automated systems. Development of the Mentor hypertext Electronic Performance Support System can help meet this need for cost-effective CAD training done within the context of current working projects, by incorporating embedded online instruction into MicroStation CAD software.

This study enhanced the Mentor CAI system for use with MicroStation software and tested the system with users representing a range of disciplines and CAD expertise. At the end of the test period, a survey was administered to determine their use of the system, attitudes, learning preferences, and reactions to the program.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1995	Final

4. TITLE AND SUBTITLE
Effectiveness of an Electronic Performance Support System for Use With Computer-Aided Design Software

5. FUNDING NUMBERS
4A162784
AT41
FF-AQ4

6. AUTHOR(S)
Doris S. Shaw, Thomas Mack, Julie L. Webster, Daniel J. Nix, and Donald K. Hicks

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Construction Engineering Research Laboratories (USACERL)
P.O. Box 9005
Champaign, IL 61826-9005

8. PERFORMING ORGANIZATION REPORT NUMBER
FF-95/06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Headquarters, U.S. Army Corps of Engineers
ATTN: CEMP-ES
20 Massachussetts Avenue, NW.
Washington, DC 20314-1000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Although Bentley Systems, Inc. (BSI) CAD, a computer-aided design program used in many U.S. Army Corps of Engineers offices, can produce the detailed documents necessary for planning, review, construction, and management of a facility, such software requires extensive training. Allocating the time and resources to learn such computer-based technology is a pressing productivity concern in both government and commercial settings. The U.S. Army Construction Engineering Research Laboratories (USACERL) has studied the use of Computer-Aided Instruction (CAI) to train architects and engineers to use many types of automated systems. Development of the Mentor hypertext Electronic Performance Support System can help meet this need for cost-effective CAD training done within the context of current working projects, by incorporating embedded online instruction into MicroStation CAD software.

This study enhanced the Mentor CAI system for use with MicroStation software and tested the system with users representing a range of disciplines and CAD expertise. At the end of the test period, a survey was administered to determine their use of the system, attitudes, learning preferences, and reactions to the program.

14. SUBJECT TERMS
computer aided instruction
computer aided design
electronic performance support system
Mentor CAI system

15. NUMBER OF PAGES
72

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Foreword

This study was conducted for Headquarters, U.S. Army Corps of Engineers under Project 4A162784AT41, "Military Facilities Engineering Technology"; Work Unit FF-AQ4, "Adaptive Workstation Environment." The technical monitor was Hugh Adams, CEMP-ES.

The work was performed by the Facility Management Division (FF) of the Infrastructure Laboratory (FL), U.S. Army Construction Engineering Research Laboratories (USACERL). Alan Moore is Chief, CECER-FF, and Dr. David M. Joncich is Acting Chief, CECER-FL. The USACERL technical editor was William J. Wolfe, Information Management Office.

LTC David J. Rehbein is Commander and Acting Director of USACERL, and Dr. Michael J. O'Connor is Technical Director.

Contents

List of Figures and Tables

Figures

Tables

1 Introduction

Background

Computer Aided Design (CAD) software is the most commonly used automated tool in the design environment. Most widely used CAD systems—like Bentley Systems, Inc. MicroStation CAD, used in many U.S. Army Corps of Engineers' offices—can produce the detailed documents necessary for planning, review, construction, and management of a facility. However, users of CAD software require extensive training. Allocating the time and resources to learn such computer-based technology is a pressing productivity concern for design professionals in both government and commercial settings. Professional training in automated technology has been estimated to cost as much as $5000.00 per week including tuition, travel, per diem costs, and labor plus overhead. Over time, that cost multiplies as vendors release new software versions and as new applications are developed.

An appealing solution to the problem of high training costs is to use the computer itself to conduct the needed instruction. The U.S. Army Construction Engineering Research Laboratories (USACERL) has studied the use of Computer-Aided Instruction (CAI) to train architects and engineers to use many types of automated systems. One USACERL effort developed a tutorial embedded in AutoCAD (a widely used CAD system) that used the display and programming capabilities of the CAD system to deliver instruction, provide exercises, measure success, and administer feedback. On a test of the basic commands, users of this system scored as highly as classroom-taught designers (Shaw and Golish 1988).

Follow-up interviews and questionnaires also revealed that, after a short period of familiarization with the system, many prefer to learn CAD as they work, within the context of their projects (Shaw and Golish 1989). Users specifically expressed a a need for help with software updates, review of forgotten material, and additional resources in their continuing use of the CAD system. Development of the Mentor CAI system, a hypertext Electronic Performance Support System (EPSS), can help meet this dual need, for cost-effective CAD training done within the context of current working projects, by incorporating embedded online instruction into MicroStation software.

Objectives

The overall objective of this study was to help reduce the cost of training end-users of MicroStation software by creating a CAI system to substitute for more expensive, time-consuming, formalized classroom instruction. Specific objectives of this part of the study were to improve the Mentor CAI system for use with MicroStation software, to observe the use of the Mentor system in the field, to analyze the way users interfaced with Mentor, and to recommend further improvements to the system. A secondary objective was to refine the feedback system itself, to improve subsequent testing of this and other CAI systems.

Approach

A literature search was done to formulate a sound theoretical basis for an automated instruction (CAI) system to accompany CAD software (Chapter 2). A hypertext Electronic Performance Support System (EPSS) that operates concurrently with MicroStation software that is widely used by the Corps of Engineers, was developed and coded (Chapter 3). The effectiveness of the system was tested with users of the CAD system at a number of volunteer test sites for a period of 20 working days. USACERL researchers visited three test sites to deliver the software and to explain the test to a point of contact at the site, who was made responsible for administering the materials. The remaining sites received materials and instructions by mail. At the end of the 20-day test period, participants responded to a 30-question survey to gauge their use of the system, attitudes, learning preferences, and reactions to the system. Results of the surveys were tabulated and analyzed, and recommendations were formulated to help improve this and future CAI systems, and also to refine the feedback and testing process (Chapters 3 and 4).

Scope

This study measured the use of, and users' reactions to the use of the Mentor EPSS system in the work environment. This study did not attempt to evaluate the influence of the Mentor system on the quality or quantity of CAD drawings produced.

Mode of Technology Transfer

The initial performance support system, the "MicroStation Mentor," was fielded under a Cooperative Research and Development Agreement (CRDA) with Intergraph

Corporation. A new CRDA with Electronic Courseware Systems (ECS) will co-develop and market a multi-media, three-dimensional (3D), PSS based on concepts investigated in the field test of the MicroStation Mentor.

2 Theoretical Basis for Electronic Performance Support System

Early CAI

Even though educational technology is a very young science, there are some theories that have affected its development. CAI is an outgrowth of the "programmed instruction movement" credited to B.F. Skinner in the late 1950s (Skinner 1958). This approach was based on the deductive reasoning paradigm known as "behaviorism," which dominated the educational world for over 30 years. Behaviorist theories held that knowledge was objective and its structure was independent from a particular context. This implied that learning could be sequenced, measured, and quantified according to the symbolic structure of any subject domain. Instructional design became product-oriented and concerned with a top-down representation of knowledge and memory aids. Since computers were able to deliver instruction and measure responses, the major problem for programmed learning became one of determining the correct order of the "teaching frames" (Markle 1978). According to this philosophy, each screen should be designed as an extension of the previous one and the steps should be small, allowing little chance of student failure.

Learning Styles

When it became apparent that some students did not profit as much as others from programmed lessons even if the lessons appeared to be well-designed, research in learning style as it related to CAI became essential. "Discovery" and "structured" learning styles as researched by Mary Stoddard at Los Alamos (1985) shaped USACERL research on CAI tutorial for CAD (Shaw and Golish 1988). According to Stoddard, discovery-oriented learners prefer to direct their own learning, as opposed to structured learners, who wish to have their learning controlled by teachers or computers. This study assumed that CAI could be made suitable for both learning styles. The USACERL embedded tutorial offered a sequential, computer-controlled format with specific practice exercises for learning the CAD commands and applications, but also permitted exploration as the student desired. The term "guided discovery learners" describes students who wanted the computer to control the order of study at times, but also wanted to occasionally experiment under their own

guidance. This type of learner seemed to be more successful at learning from this CAI. Discovery learners appeared to leave out much essential learning and be less efficient while structured learners often wanted even more structure than the computer alone could offer.

In many cases, learning styles seemed to change as students became more familiar with the system. Typically, students who were quite "structured" when they began the lessons would change in the direction of "discovery" learning as they became more familiar with the system and the method of work. Embedded training in a CAD system, in which all the power and complications of the system were available to the student, seemed to frighten many beginners. This is similar to findings in other subject areas that concern "fidelity" of the learning environment to the work situation (Alessi 1988), such as flight simulation or emergency medical treatment. Studies showed that the fear element was sufficiently strong as to deter people from ever flying or nursing. It was theorized that, as the students gained confidence, they would begin to feel free to test ideas about how the system worked and to experiment on individual interests. The matter of "locus of control" as it relates to experience in the system, learning style, and the task became a main concern as well as the subject of examination by other CAI researchers.

User Control

Nelson (1974) proposed "computer-based presentational wonderlands, where a student (or other user) may browse and ramble through a vast variety of writings, pictures and apparitions." This is not the usual type of CAI, but a resource rich environment in which students can develop abilities to "think, argue and disagree intelligently." Students are empowered to follow their own paths to ideas and solutions. Nelson's concepts suggest that hypertext and hypermedia approaches should replace the machine-controlled tutorial or dialog style in CAI. Through "hyper" systems, students are given the responsibility of guiding their own learning.

Another system directed at the way computers can "contribute to mental processes" is the Logo microworld of Seymour Papert (1980). In this system, students are permitted to construct and debug procedures to explore "the power of ideas and the power of the mind." Logo is another example of a user-controlled computer environment aimed at developing individual thinking. Often, user-controlled systems were found to be unsuccessful in practice. In a review of studies on the subject, students were found to be deficient in determining what and how much to study (Steinberg, 1977). Steinberg (1989) later reported that some type of structure was helpful to learners who were not familiar with the subject matter or the system, but that

sophisticated learners performed well in user-controlled environments. All this quantitative comparison has not yet resulted in a new theory to build on, but it has pointed out the deficiencies in that kind of paradigm for research (Reeves 1993). Properly conceived case studies and ethnographic methods can enrich the understanding and prevent tunnel vision.

Cognitive Theories

In this research, after the initial, more timid phase of learning, users developed a wide variety of special expertise that seemed to be directly related to their task orientation. This can be expected according to theories of context-based, situated learning in cognitive science (Brown, Collins, and Duguid 1989). Cognitive theories propose that the learning environment is constructed by the learner according to individual patterns that are inductive, bottom up, and process-oriented. This study was aimed at learning in the real world in ways that would effectively support specific tasks, previous states of knowledge, and preferences. Success in that goal requires an understanding of the factors that lead to the ability to acquire knowledge in complex domains (Spiro, et al. 1992).

Recent constructivist theories emphasize a relationship of experience to understanding that contrasts with the context-free, behaviorist views. Highly organized, sequential instructional programs have been questioned. For one thing, that practice is conducive to oversimplification. Initial over-simplification of complex subject domains has been shown to have a handicapping effect on advanced concept development (Spiro, Jacobson, and Jehng 1988). Learners do not seem to develop the parameters for increasing complexity when the environment is created and controlled for them. Another problem is that determining a proper sequence depends on the student's priorities. One person, for example, may want to draw a line on the screen and worry about precision later while another would not be comfortable drawing a line unless it was precise. There is no clear imperative, whether to teach "draw line" or "precision placement" first. Another important consideration is the difficulty of incorporating new material into sequential presentations. The modern information explosion makes it more difficult to define the limits for any learning context. New CAI approaches that encompass cognitive theories are being developed and tested to meet these challenges.

Electronic Performance Support Systems

The concept of EPSS has been described by Gloria Gery (1991) with several real-world examples of the type of program that is the subject of this research. Hypertext links with a variety of different approaches that the learner can be navigate as desired makes up the basic system requirements. Multi-media, user monitoring, adaptive systems, active assistance, and networked resources (Modesitt 1994) are only a few of the technologies that can be incorporated into EPSS. The design counters the problems of incremental sequencing, oversimplification, machine control, and extensibility found in traditional CAI. The systems offer great variability; the power that can be put into the hands of students increases daily.

The theories that will shape the development of CAI in the future are still in development. Technological advancement appears to be a kind of evolution. It can be represented as the creation of needed artifacts that work (Tripp 1992). The pragmatic test is the marketplace. Theory often follows initial application development, as in the case of bridges that were built before structural theories that served as models to derive theories. It was said that "Physics came from the steam engine, not vice versa" (Basalla 1988). This study hopes to contribute to the development of EPSS theory.

A few theories have been tested in the Area of CAI. Hick's Law of choice reaction times, Fitt's Law dealing with the time it takes to move the hand to a target, and the Power Law of Practice that predicts keystroking performance (Card, Moran, and Newell 1983) are interesting theories generally related to this area of study, but contribute little more to this specific domain beyond screen design. This research requires theories that can help to design CAI that can motivate students, permit them to relate the information to their previous knowledge, provide the right sort of flexibility, and, as stated in an interview with the eminent cognitive scientist, Donald Norman, "make the computer invisible, so that you really have a device which helps you do your task" (Fishman and Silver 1990).

3 Technical Development

Development of the Mentor

The prototype of the Mentor, implemented in MicroStation 3.3, operated through User Commands, a primitive scripting language that offered the only programming then available in MicroStation. The revised Mentor was developed using MicroStation Development Language (MDL). MDL, which became available in version 4.0, is a powerful but complicated C-like code executed by MicroStation that offers the ability to create seamless add-on applications to MicroStation. MDL is similar to the C programming language in structure, therefore a knowledge of C helps in developing MDL programs. The Mentor has relatively simple programming so that an understanding of its basic MDL structure can allow the development of powerful educational software. Before discussing the MDL programming specifics, one needs a basic understanding of the Mentor's organization.

Organization of the Mentor

The Mentor is arranged in a hierarchical format. This hierarchy allows the user easy access to information without having to scroll through large amounts of irrelevant

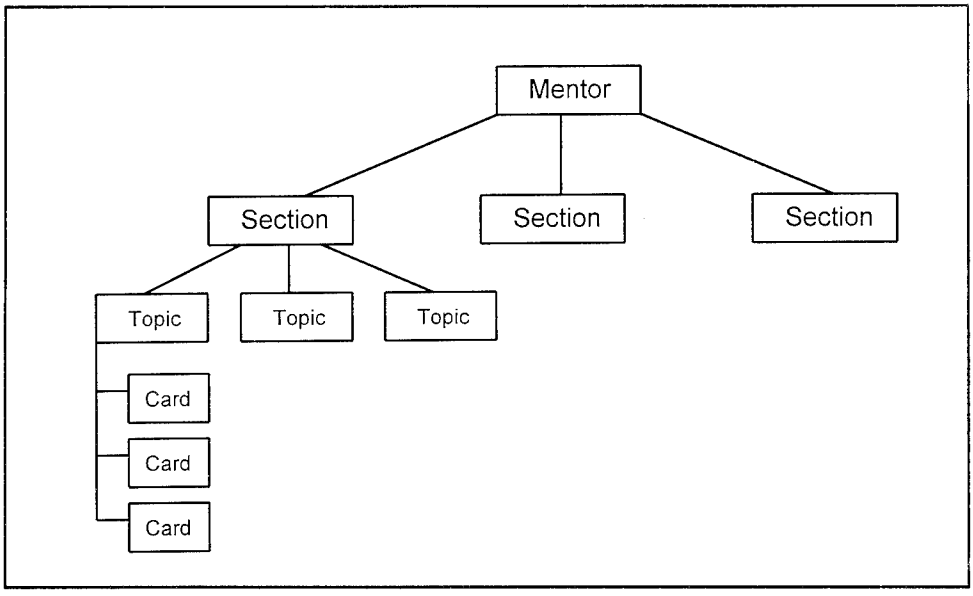


Figure 1. Schematic of Mentor hierarchy.

information. Figure 1 shows the Mentor hierarchy expressed in the form of pull-down menus accessed from the Mentor's control box.

The Mentor is divided into sections (synonymous with the chapters of a book) divided into topics that contain "information cards." Each card contains only a small portion of information. Each "chunk" of information links to separate but relevant pieces of information via hot keys and push buttons.

Mentor Components

Cards

The card, the basic building block of the Mentor, is simply a MicroStation dialog box. It is the media that delivers the information. The Mentor is comprised of about 450 dialog boxes, one dialog box for each card. Each dialog box contains the items needed to make it a card. Each card contains text and some of the cards contain pictures and/or icons for illustration. All of the cards contain push buttons that allow the user to move to other cards.

Because the Mentor is developed with MDL, many of the routines needed to drive the program already exist. For example, the routines needed to open, draw, and operate all dialog boxes in MicroStation are available and readily accessible through MDL. The dialog boxes used in the Mentor are the same boxes used by MicroStation. Many of the items used in Mentor dialog boxes are standard MDL dialog box items. MicroStation controls drawing, placement, and most of the operation of these items. Standard items include push buttons, pull-down menus, group boxes, and text. Other nonstandard items (called "generic items") are used in some Mentor dialog boxes. Generic items in the Mentor display pictures and icons in the cards. Figure 2 shows typical items included in a Mentor dialog box.

Push Buttons

Each Mentor card contains a collection of push buttons at the bottom of the card. Each of these buttons sends the MicroStation user to another card. Pushing a button issues a command that closes the current dialog box and opens the dialog box that corresponds to that button. Commands are discussed later. For every Mentor card, there is only one push button that opens each dialog box. The button is used repeatedly to link Mentor cards together. Since the push button is a standard dialog box item, MDL and MicroStation define most of its operation.

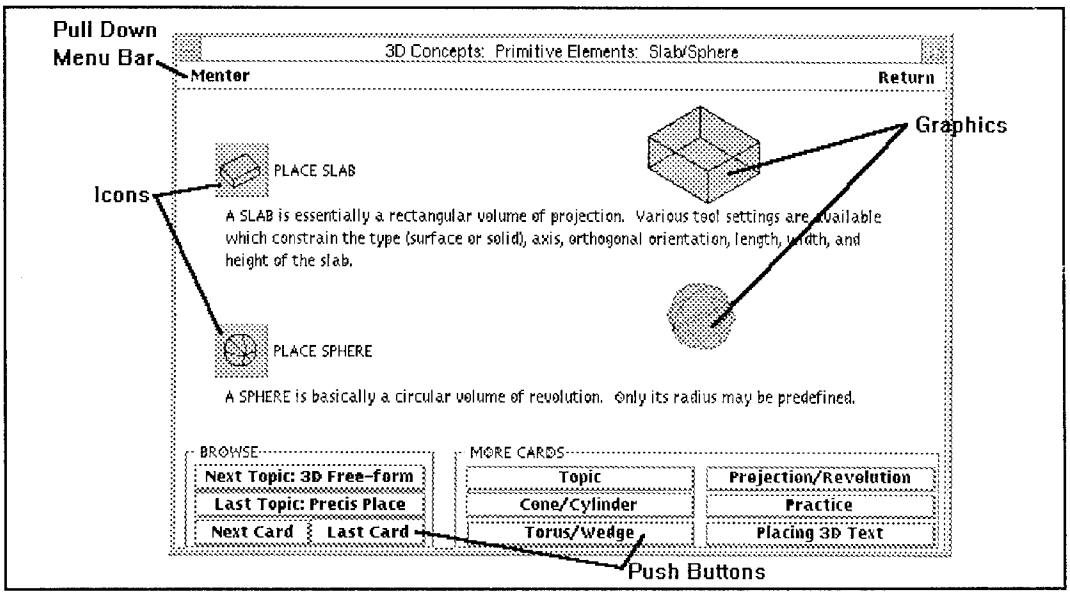


Figure 2. Typical Mentor card or dialog box.

Understanding how the buttons are used and how they link cards is necessary to understanding how the Mentor works. The following example shows a simplified version of the Mentor. Assume there are four cards named Card 1, Card 2, Card 3, and Card 4. In the Mentor, there is one button for each card, therefore this example has 4 buttons named Button 1, Button 2, Button 3, and Button 4. Button 1 opens Card 1, Button 2 opens Card 2, and so on. Notice in Figure 3 that Button 1 appears in Cards 2, 3, and 4. This is the same button used three times. In fact, all of the buttons are used three times. Each card has three buttons to create three links to three other cards. The Mentor is arranged similarly. Buttons create links to other cards.

Pull-Down Menus

Pull-down Menus are contained in a standard dialog box item called a menu bar. The menu bar contains the pull-down menus, which contain either sub-menus or commands. A sub-menu is a secondary menu that branches off the primary pull-down menu. When no sub-menu exists, the pull-down menu item will execute a command similar to a push button. The command closes the current Mentor dialog box and opens the Mentor dialog box corresponding to the pull-down menu item. The Mentor has only one menu bar that is used repeatedly in every dialog box.

Commands

Similar to push buttons and pull-down menu items, there is exactly one command for each Mentor card. Recall that the push buttons and pull-down menu items execute

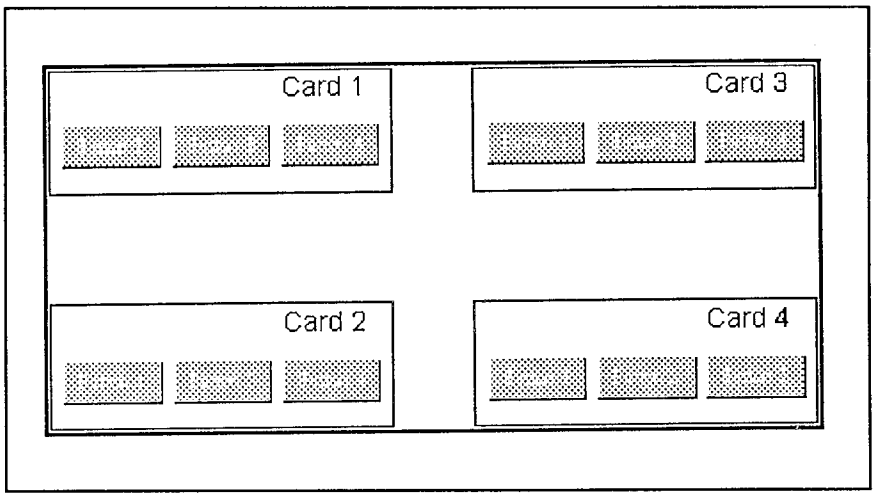


Figure 3. Simplified model of the Mentor.

commands. Actually a push button and a pull-down menu item that open a particular dialog box execute the same command. Commands may be executed by several different items. In the Mentor, commands are primarily executed in three ways: push button items, pull-down menu items, or key-ins (straightforward keyboard entries).

Mentor commands are rather simple. They basically perform two functions: (1) close any existing Mentor card dialog boxes, and (2) open the dialog box requested by the user. Figure 4 models the following operation. The first event: (1) issues the command, which (2) closes any existing dialog boxes and opens the requested dialog box. (This sequence is programmed for each card in the Mentor.) For example, Push Button 1 executes Command 1, which opens Card 1. Also, Pull-down Menu Item 1 or Key In 1 executes Command 1, which opens Card 1. The Mentor is a collection of such command sequences.

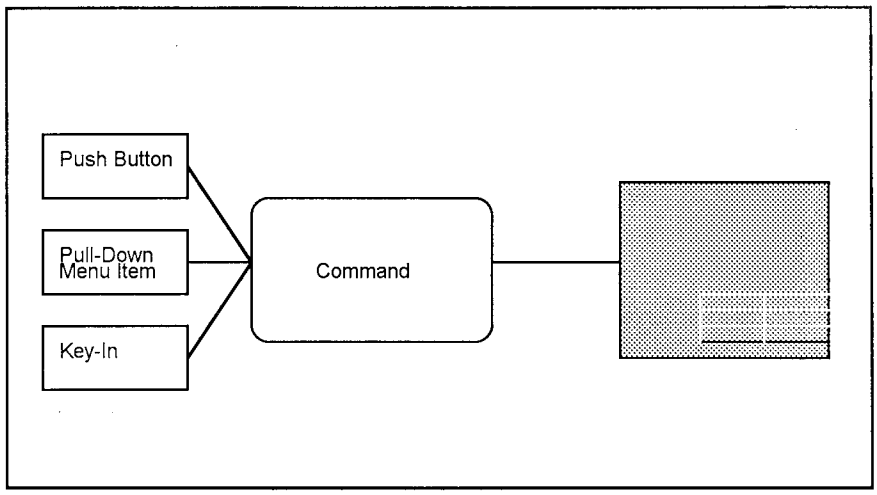


Figure 4. Typical command sequence in the Mentor.

MDL Programming

Understanding the development of the Mentor requires a basic understanding of C programming and MDL. The Mentor follows classic MDL structure. Included at the end of this section are some suggested readings that will make the Mentor development process more understandable.

File Structure and Development Tools

Many files combine to make the Mentor. Each file has an extension that indicates the file type. Table 1 shows the file conventions used by the Mentor and most MDL developers.

MDL has several tools used in application development. They include compilers and linkers that create the final (*.ma) application file. Table 2 summarizes the development tools used to create the Mentor. These tools are included with MicroStation. The suggested readings at the end of this chapter can provide more information about the development tools.

Figure 5 shows the Mentor development process. The bmake command automates this process, greatly simplifying the procedure. After making changes to the program, bmake automatically runs the entire linking and compilation operation.

Table 1. MDL file conventions.

File Extension	File Description
*.h	Header or include file. These are similar to regular C header files. They contain definitions used by multiple source files.
*.ma	MDL application file. The application is loaded into MicroStation and run with the MDL LOAD command.
*.mc	C like source code for an MDL application. The Mentor has three C like source files: the commands.mc, hooks.mc and main.mc.
*.mke	Make file used to automate the entire compile process. See the MDL programmer's manual under BMAKE.
*.mo	Compiled .mc file. This is an intermediate file created during the BMAKE process.
*.mp	Intermediate file used in the BMAKE process. Referred to as a Program file.
*.mt	Declares a global sructure that is seen by all files.
*.r	Resource source file. Used to define dialog boxes, dialog box items, command tables, etc. Most of the Mentor code is located in these file types.
*.rsc	Compiled resource file.

Table 2. MDL development tools.

Tool	Description
mcomp	Compiles MDL source code (*.mc) to object files (.mo).
mlink	Links object files (*.mo) to program files (.mp).
rcomp(-h)	Compiles resource source files (*.r) to resource files (*.rsc).-h option used to make a header file from command table.
rlib	Links program files (*.mp) and resource files (*.rsc) to application files (*.ma).
bmake	Automates all the steps required to make an application.

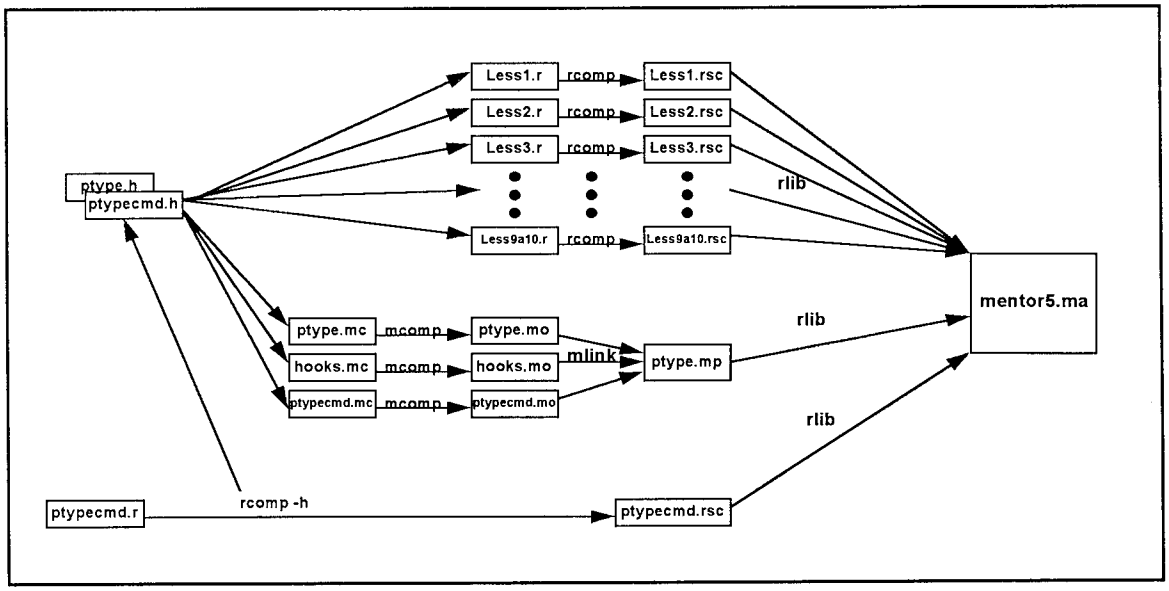


Figure 5. Development cycle of the MicroStation Mentor.

Header (Include) File

Header (or Include) files in MDL are the same as header files used in regular C programs. Those familiar with the usage of header files will quickly learn to use header files in the Mentor and MDL.

Every dialog box, push button, pull-down menu item, etc. is assigned a unique number as an ID. For example, there is only one push button with ID number 1. Similarly, there is only one dialog box with the ID number 1. To avoid the possibility of replicating numbers, it is best to keep ID numbers separate in a header file. Each card in the Mentor is given a title that is shown in its title bar. It is easier to keep track of the Mentor card titles than the many ID numbers associated with items in the Mentor. For example, Figure 6 shows the IDs of dialog boxes used for the Guide portion of the Mentor.

Other items in the Mentor have their unique ID declared in the header file as well. Examples include #define PUSHBUTTONID Guide Title 158 and #define PULLDOWNMENU-ID Guide 767. Again, placement of the IDs in the header file ensures that the item numbers used in the Mentor remain unique.

#define DIALOGID Guide Title	622
#define DIALOGID Guide Intro	623
#define DIALOGID Guide Cntrl Topic	624
#define DIALOGID Guide Cntrl More	625
#define DIALOGID Guide Cntrl NU	626
#define DIALOGID Guide Cntrl Update	627
#define DIALOGID Guide Cntrl Lookup	628
#define DIALOGID Guide Cntrl Pict	629
#define DIALOGID Guide Inp Topic	630
#define DIALOGID Guide Inp Mouse	631
#define DIALOGID Guide Inp Dig	632
#define DIALOGID Guide Cont Topic	633
#define DIALOGID Guide Cont More	634
#define DIALOGID Guide Sum	635

Figure 6. Section of a Mentor header file.

Resource Source File

Since the Mentor is a collection of dialog boxes, the resource source files (*.r) contain most of the Mentor's code. The resource files contain items called "resources" that create the dialog boxes, push buttons, pull-down menus, etc. There are many different types of resources in MDL. Those resources that define dialog boxes are called dialog box resources. Resources that define items in dialog boxes are called item resources. All resources available in MDL are similar in form and function. Each has particular fields that control different attributes. Information defined in a dialog box resource includes: type and size of dialog box, text on title bar, and items included in the dialog box (Figure 7).

Push Buttons

The dialog box resource contains a list of items included in the dialog box, called "item instances." Item instances tell MicroStation what to put into a dialog box. Mentor items include push buttons, text, a menu bar (pull-down menus), and generic items that are nonstandard dialog box items. The MDL programmers guide gives more information about standard dialog box items. Figure 8 shows an item instance from the dialog box resource in Figure 7.

Information included in an item instance are: location and width of the item in the dialog box, item type, item ID, and some additional information. Table 3 explains each field. Just as every dialog box has dialog resources, every item has item resources. Figure 9 shows the item resource for the item instance shown in Figure 8. The unique IDs link the item instances and the item resources.

```
DialogBoxRsc DIALOGID  Guide  Intro =
{
DIALOGATTR  SINKABLE,/*Type of  Dialog Box*/
80*XC, 25*YC,/*Size of Dialog Box*/
NOHELP,MHELP,HOOKDIALOGID  MMentorBoxAction, NOPARENTID,/*More Attributes*/
"Guide:  Introduction",/*Text on Title Bar*/
{
/*Begin of Item List*/
{{0,0,0,0}, MenuBar, MENUBARID  Primary,ON,0,"","},
{{ XC,BrwsBoxY, BrwsBoxWth, BrwsBoxHgt }, GroupBox, 0,ON, 0, "BROWSE",""},
{{ NCardX,NCardY,NCardWth,NCardHgt}, PushButton, PBUTTONID  Guide  Cntrl  Topic,

     ON, 0, "Next Card", ""},
{{ NTopX,NTopY, PBstdWth, PBstdHgt}, PushButton, PBUTTONID  Guide  Cntrl  Topic,

     ON, 0, "Next Topic: Controls", ""},
{{MCardBoxX, MCardBoxY, MCardBoxWth, MCardBoxHgt}, GroupBox,0,ON,0, "MORE CARDS",""},
{{LABORGX,LABORGY,LABEXTX,LABEXTY},
Label,BULKTEXTID,ON,ALIGN  LEFT|LABEL  WORDWRAP,
"\n\
 The Guide to MicroStation Mentor will explain \
how to navigate through the dialog boxes \
of the Mentor Performance Support System.  \
These dialog boxes are often called cards.\n\
\n\
It will also provide illustrations that will help \
to select the type of support that will be most efficient \
for each individual, a brief introduction to the contents \
of the Mentor, and a short description of input devices.\n\
\n\
One way to select a different Mentor card is: \n\
\tClick on the Mentor pull-down menu in the upper left \
corner of this card. \n\
\tThen select the topic of choice. \n\
The way to see the next card in the Mentor sequence is: \n\
\tClick on Next Card in the Browse box below. ",""},
}
};
```

Figure 7. Sample dialog box resource.

The item resource defines more specific information about what the item instance does. The item instance describes the item's placement in the dialog box and the item resource tells MicroStation what action the item performs. Table 4 describes each field in the item resource shown in Figure 9.

Recall that dialog box resource defines a dialog box. It contains a list of item instances that define items to included in a dialog box. Item instances describe the placement of items in the dialog box (i.e., location, size, and type). The item instances each have an item resource that defines what the item does. In the case of push buttons, the item resource tells MicroStation the specific command to activate. Up until now, discussion on the resource files has concentrated on push buttons. Other items contained in the Mentor cards include text and pull-down menus.

```
{{NCardX,NCardY,NCardWth,NCardHgt},PushButton,PBUTTONID  Guide  Cntrl  Topic,
    ON, 0, "Next Card", ""},
```

Figure 8. Example item instance.

Table 3. Description of item instance.

Field	Meaning
NCardX	Defines the X dialog coordinate of the push button. NCardX is defined in a header file because the "Next Card" button is in the same location on every card.
NCardY	Defines the Y dialog coordinate of the push button. Defined in a header file.
NCardWth	Defines width of push button. Defined in a header file.
NCardHgt	Defines height of push button. Defined in a header file.
PushButton	Define the item type in the dialog box.
PBUTTONID Guide Cntrl Topic	Unique push button ID number. Defined in header file.
ON	Enables the button.
0	This field not used for push buttons.
"Next Card"	Override text placed onto button.
""	Override access string (not used).

Pull-down Menus

The pull-down menus are encapsulated in a standard dialog box item called a menu bar. The dialog box resource in Figure 7 contains an item instance for the Mentor menu bar (Figure 10).

This item instance appears in every dialog box resource. Since there are approximately 450 dialog boxes in the Mentor, this item has approximately 450 instances. Table 5 describes each field of the menu bar item instance shown in Figure 10.

The unique ID links the menu bar item instance to the menu bar item resource. Figure 11 and Table 6 show and describe the menu bar item resource. Notice that both the item instance and the item resource contain the ID MENUBARID Primary. The menu bar item resource is more complex than the push button resource because it contains item instances similar to the dialog box resource. In the Mentor, the menu bar contains two pull-down menus, one titled "Mentor" and the other "Return." (The MDL Programmers Guide gives more information about other items to include in menu bars.)

```
DItem PushButtonRsc PBUTTONID Guide Cntrl Topic =
{
NOT DEFAULT BUTTON, NOHELP, MHELP, HOOKITEMID Button StandardAction,
NOARG,
CMD MEN GUIDE CONTROLS TOPIC,
LCMD,
"",
"Topic"
};
```

Figure 9. Example push button item resource.

Table 4. Description of fields in push button item resource.

Term	Description
DItem PushButtonRsc	Describes the type of resource. This is a push button resource.
PBUTTONID Guide Cntrl Topic	Unique push button ID. Defined in the header file.
NOT DEFAULT BUTTON	Tells BSI CAD that this is not the default button. See MDL programmers guide for other options in field.
NOHELP	No help for this item.
MHELP	If there was help, it would be MicroStation help.
HOOKITEMID Button StandardAction	This is a hook function that is linked to the item. More on hook functions later.
NOARG	No arguments are passed with this hook function.
CMD MEN GUIDE CONTROLS TOPIC	Command activated with this item.
LCMD	"Local command" owned by the Mentor. If command mentioned in previous field is a MicroStation command, use MCMD.
""	Access string (not used).
"Topic"	Default text that is placed in the button. (May be overridden by the item instance definition).

As with other resources, the menu bar resource contains fields to control attributes and item instances. As with other item instances, the pull-down menu instances have associated resources. Figure 12 shows the Mentor pull-down menu resource and Table 7 describes the first few lines of the pull-down menu resource.

The remaining part of the menu bar resource is a list of items included in the pull-down menu. Figure 13 and Table 8 show a single item in the pull-down menu and the meaning of each field. This sub-menu has item resources like those shown in Figure 12. Most pull-down menus in the Mentor are text pull-downs, which means that the pull-down contains a series of text strings selections.

```
{{0,0,0,0}, MenuBar, MENUBARID  Primary,ON,0,"","},
```

Figure 10. Example menu bar item instance.

Table 5. Description of menu bar item instance.

Field	Description
{0,0,0,0}	Usually defines location and size of dialog box item (but the menu bar can only appear across the top of dialog box, so this field is ignored.)
MenuBar	Type of item.
MENBARID Primary	Unique ID of item.
ON	Enables menu bar.
0	Menu bar does not pass any value so item argument is set to zero.
""	Field is unused, so set to "".

If there is no sub-menu, then an item will look as shown is Figure 14. This is similar to the instance described above, but the sub-menu type is NOSUBMENU and the pull-down ID is set to zero. Since this item has no sub-menu, it will execute the command with ID CMD MEN UNLOAD.

Text Items

Item Instance Definition. Most of the text in the Mentor cards occurs as a text item instance. The text has no item resource, only an item instance. This makes the text item different than other standard dialog box items. Figure 15 contains the text item instance for the Guide Introduction dialog box. Table 9 lists a description of the label text item terms. Since text executes no commands and alters no variables, there is no need for an item resource.

The text item has one distinction that needs clarification: the ID of the item is not unique. Every main text item in the Mentor has the ID BULKTEXTID, therefore the ID field is no longer needed. The ID remains from previous prototypes of the Mentor and can be set to zero. Remember that the ID links item instances to item resources. Since the text item has no resource, the ID is not needed.

```
DItem  MenuBarRsc MENUBARID  Primary =
    {NOHOOK,NOARG,
        {/*Begin list of items instances in Menu Bar*/
        {PulldownMenu,PULLDOWNMENUID  Mentor  Main},
        {PulldownMenu,PULLDOWNMENUID  Return},
/*End List*/
        }
    };
```

Figure 11. Example menu bar item resource.

Table 6. Description of menu bar item resource.

Field	Description	
DItem MenuBarRsc	Defines the type of resource.	
MENUBARID Primary	Unique ID.	
NOHOOK	No hook function defined to menu bar.	
NOARG	No argument passed with menu bar.	
Menu Bar Item Instances	PulldownMenu	Place pull-down menu in menu bar.
	PULLDOWNMENUID Mentor Main	Pull-down menu unique ID.
	PulldownMenu	Place pull-down menu in menu bar.
	PULLDOWNMENUID Return	Pull-down menu unique ID.

Formatting of Text in a Text Item. Mentor formatting codes are similar to regular C printf format codes. Each line must end with a backlash (\). This tells MicroStation that the text continues on to the next line. If the line does not end with a backlash, an error will occur. Other formatting codes are available for carriage returns and tabs. Table 10 shows formatting codes used in the Mentor.

Hot Keys

Hyperlinks. On some Mentor cards, the text contains hot keys that create "hyperlinks" to other cards. Remember that push buttons create links to other cards. Also, push button item instances contain an override field for text. If the override field is blank, the default text defined in the item resource is used on the button.

The simple dialog box shown in Figure 16 contains two instances of the same button. Figure 17 contains excerpts of the resource file used to create the dialog box and buttons. Notice similarities in the item instances. They have the same ID of PBUTTONID Card1, which indicates that both instances use the same resource with

```
DItem PulldownMenuRsc PULLDOWNMENUID Mentor Main =
{NOHELP, MHELP, NOHOOK, ON | ALIGN LEFT, "Mentor",
  {
{"Guide", NOACCEL, ON, NOMARK,PulldownMenu,
  PULLDOWNMENUID Guide,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN GUIDE TITLE,LCMD,""},
{"Simple Elements", NOACCEL, ON, NOMARK,PulldownMenu,
  PULLDOWNMENUID Less1,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN LESSON1 TOPIC,LCMD,""},
{"Complex Elements", NOACCEL, ON, NOMARK,PulldownMenu,
  PULLDOWNMENUID Less2,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN LESSON2 TOPIC,LCMD,""},
{"Element Manipulation", NOACCEL, ON, NOMARK,PulldownMenu,
  PULLDOWNMENUID Less3,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN LESSON3 TOPIC,LCMD,""},
{"3D Concepts", NOACCEL, ON, NOMARK,PulldownMenu,
  PULLDOWNMENUID Less7,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN LESSON7 TOPIC,LCMD,""},
{"Architectural", NOACCEL, ON, NOMARK,PulldownMenu,
  PULLDOWNMENUID Less10,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN ARCHITECTURAL TOPIC,LCMD,""},
{"Unload Mentor", NOACCEL, ON, NOMARK,NOSUBMENU,
  0,NOHELP,MHELP,
  NOHOOK, 0,
  CMD MEN UNLOAD,LCMD,""},
}
  };
```

Figure 12. Example pull-down menu resource.

Table 7. Description of upper portion of pull-down menu resource.

Field	Description
DItem PulldownMenuRsc	This pull-down menu resource.
PULLDOWNMENUID Mentor Main	Unique ID of Pull-down Menu.
NOHELP	No help associated with this item.
MHELP	If their was help it would be MicroStation help.
NOHOOK	No hook function associated with this item.
ON | ALIGN LEFT	MDL allows options to be combined with the pipe (|). This says that the menu is enabled and left justified.
Mentor	This is the text that is placed on the menu bar. The title of the pull-down menu.

PBUTTONID Card1. Differences in the item instances include the location of the buttons and text in the override label field. The hot key button contains the text "hot key." This overrides default text "Card 2." Both buttons execute the same command and open the same dialog box.

```
{"Guide", NOACCEL, ON, NOMARK,PulldownMenu,
          PULLDOWNMENUID_Guide,NOHELP,MHELP,
          NOHOOK, 0,
          CMD_MEN_GUIDE_TITLE,LCMD,""},
```

Figure 13. Example item in pull-down menu resource.

Table 8. Description of first pull-down menu item.

Field	Description
"Guide"	Title of menu.
NOACCEL	No accelerator key code.
ON	Enables the pull-down menu.
NOMARK	No type of mark displayed on menu. (MDL programmers guide gives other options.)
PulldownMenu	Sub-menu type field. This specifies the pull-down sub-menu.
PULLDOWNMENUID_Guide	Unique ID of sub-menu. Usually ID to another pull-down menu.
NOHELP	No help defined for item.
MHELP	If there was help, it would MicroStation help.
NOHOOK	No hook function defined.
0	No argument sent to hook function.
CMD_MEN_GUIDE_TITLE	ID of command executed by pull-down menu.
LCMD	"Local command" owned by Mentor.
""	Unused field.

```
{"Unload Mentor", NOACCEL, ON, NOMARK,NOSUBMENU,
    0,NOHELP,MHELP,
    NOHOOK, 0,
    CMD_MEN_UNLOAD,LCMD,""},
```

Figure 14. Example of pull-down menu item with no submenu.

View Buttons. On several Mentor cards, the text refers to saved views in the Mentor design file. To see a saved view, the user keys in "VI=NAME," where NAME is the label of the view. If the user wants to see the view and is not in the Mentor design file, most cards contain a view button labeled "SHOW VI." View buttons are push buttons that open a black dialog box, called a view box. The view box displays cells that correspond to the saved views in the Mentor design file. A push button labeled "HIDE VI" closes the view box.

```
{{LABORGX,LABORGY,LABEXTX,LABEXTY},
Label,BULKTEXTID,ON,ALIGN_LEFTILABEL_WORDWRAP,
"\n\
The Guide to MicroStation Mentor will explain \
how to navigate through the dialog boxes \
of the Mentor Performance Support System.  \
These dialog boxes are often called cards.\n\
\n\
It will also provide illustrations that will help \
to select the type of support that will be most efficient \
for each individual, a brief introduction to the contents \
of the Mentor, and a short description of input devices.\n\
\n\
One way to select a different Mentor card is: \n\
\tClick on the Mentor pull-down menu in the upper left \
corner of this card. \n\
\tThen select the topic of choice. \n\
The way to see the next card in the Mentor sequence is: \n\
\tClick on Next Card in the Browse box below. ","},
```

Figure 15. Sample text item instance.

Table 9. Description of label text item.

Field	Description
{LABORGX,LABORGY, LABEXTX,LABEXTY}	Defines item location and size. (Defined in header file.)
Label	Defines item type.
BULKTEXTID	ID of main text.
ON	Displays the text.
ALIGN_LEFTILABEL_ WORDWRAP	Left aligns the text and turns on the word wrap buffer so the text will wrap if the line exceeds the size of the text item.

Table 10. Mentor text formatting codes.

Formatting Code	Description
\	Continues text on next line. Sometimes called a "soft return." If there is room, text may continue on the current line in the card.
\n\	Advance to next line. Sometimes called a "hard return."
\t	Place tab.
\"	Place a double quotation mark

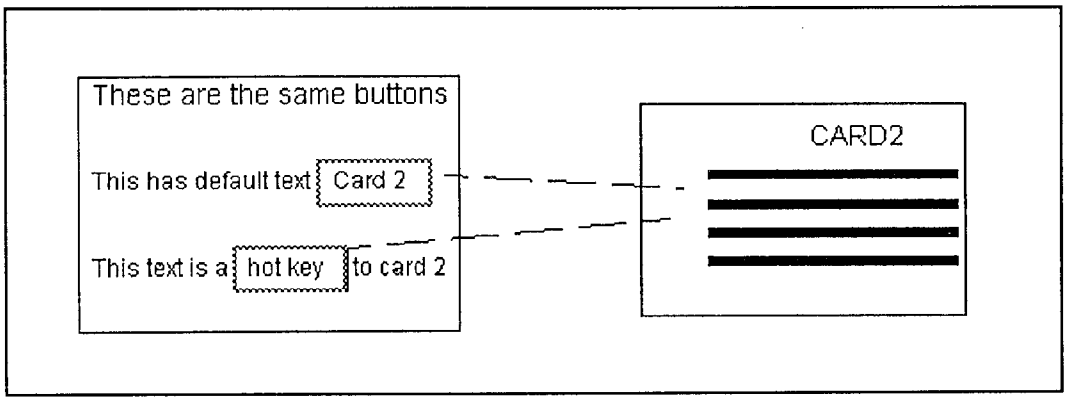


Figure 16. Dialog box with two instances of same button.

```
#include ptype.h
#include ptypecmd.h

/*::::::::::::::::::::::::::::::::::::::::::::
:: Dialog Box Resource                        ::
::::::::::::::::::::::::::::::::::::::::::::*/
DialogBoxRsc DIALOGID_Guide_Intro =
{DIALOGATTR_SINKABLE,/*Type of Dialog Box*/
40*XC, 25*YC,/*Size of Dialog Box*/
NOHELP,MHELP,NOHOOK, NOPARENTID,/*More Attributes*/" ",
{/*Beginning of Item Instance List*/
{{ NCardX,NCardY,NCardWth,NCardHgt}, PushButton, PBUTTONID_Card1,
            ON, 0, "", ""},
{{ NTopX,NTopY, PBstdWth, PBstdHgt}, PushButton, PBUTTONID_Card1,
            ON, 0, "hot key", ""},
O           O                       O           O
O           O                       O           O
O           O                       O           O
}
};
/*::::::::::::::::::::::::::::::::::::::::::::
:: Item Resource                              ::
::::::::::::::::::::::::::::::::::::::::::::*/
DItem_PushButtonRsc PBUTTONID_Card1 =
{
      NOT_DEFAULT_BUTTON, NOHELP, MHELP, HOOKITEMID_Button_StandardAction,
      NOARG,
      CMD_MEN_CARD2,
      LCMD,
      "",
      "Card 2"
};
```

Figure 17. Partial resource for dialog box shown in Figure 16.

The view button function is more complicated than the other items. Each card with a view box access will contain a push button item instance labeled "SHOW VI" (Figure 18). When the user pushes the view button, the item resource for the view button calls a hook function with ID HOOKITEMID Show VI (Figure 19) that opens the view box.

```
{{390,216,80,20}, PushButton, PBUTTONID  Show  VI, ON, 1, "", ""},
```

Figure 18. Example of item instance for view button.

```
DItem  PushButtonRsc PBUTTONID  Show  VI =
{
      NOT  DEFAULT  BUTTON, NOHELP, MHELP, HOOKITEMID  Show  VI,
      NOARG,
      0,
      LCMD,
      "",
      "Show VI"
};
```

Figure 19. View button item resource.

```
{{50,10,370,240},Generic, GENERICID  DrawCell, ON, 0, "VI",""},
```

Figure 20. Generic item to draw cell in view box.

The view box resource contains a "HIDE VI" push button and a generic item that calls a hook function to draw the cell. Figure 20 shows the generic item instance for the view box. This is very similar to the generic item to draw a cell in a Mentor card (discussed later). However, the label "VI" tells the GENERICID DrawCell hook function to draw a view box cell. The hook will retrieve the argument from the view button instance (field value 1 in Table 11). This argument will index a table of cell names designated specifically for view boxes and draw the appropriate cell.

A view cell index is necessary so the "SHOW VI" push button instance in a card (Figure 18) can reference which view cell to display. The label field (Table 11) cannot contain the cell name since the button name would automatically change to the cell name. Also, the auxiliary information field must be empty, otherwise errors occur. The argument number field is the only open field in which to place a cell reference. This field will not accept a cell name because it requires a numeric value. Therefore, the "SHOW VI" push button instance must contain an index to the view cells in the argument field.

Table 11. Description of view button instance.

Field	Description
{390,216,80,20},	Defines the button's location and size in the card.
PushButton	Item type.
PBUTTONID Show VI	Unique item ID.
ON	Enables item.
1	Argument number is 1, used to index a cell name.
""	Unused label field.
""	Unused auxiliary information.

Generic Items

In the Mentor, generic items place icons and pictures into the dialog box. Remember that generic items are not standard dialog box items, so their form and function are not pre-defined by MicroStation. The hook function associated with the generic item defines its behavior. In the Mentor there are basically two types of Generic items. One type places the icons into the cards and the other draws the pictures.

Icon Generic Item. The icon generic item places icons into the card. Icons are true MicroStation icons read from inside MicroStation. A Mentor card can show two types of icons, regular and special. A regular icon is a pre-defined MicroStation icon, such as the icons that appear on the Main palette. A special icon is an icon that is not pre-defined within MicroStation. The Mentor reads these special icons from other MDL programs. Examples of special icons are those found on 3D palettes. Figure 21 shows the generic item instance that places the Place Half Ellipse icon (a regular icon) into a Mentor card. It is very similar to other item instances previously discussed.

MicroStation assigns a unique number to each icon. MicroStation also gives each regular icon a label that represents this associated number. In Table 12, the ICONCMDID PlaceHalfEllipse attributes field represents the number of the Place Half Ellipse icon. A MDL header file (called dlogitems.h) defines the ICONCMDID's.

Both types of icons use the same GENERICID DrawIcon item resource and both place the number of the icon to display in the attributes field. The only difference between a special and regular icon's item resource is in the label field. If the label is "NONE," then the Mentor displays a regular icon. In Figure 21, the Mentor displays a pre-defined MicroStation icon whose number is ICONCMDID PlaceHalfEllipse. If the label field is not "NONE" (as in Figure 22), then it contains the name of the MDL file

```
{{4*XC,6.5*YC,8*XC,18*YC}, Generic, GENERICID_DrawIcon, ON, ICONCMDID_PlaceHalfEllipse, "NONE", ""},
```

Figure 21. Example item instance for regular icon generic item.

Table 12. Description of regular icon generic item intstance.

Field	Description
{4*XC,6.5*YC,8*XC,18*YC}	Defines the item's location and size in card.
Generic	Item type.
GENERICID_DrawIcon	ID of Item Resource.
ON	Enables item.
ICONCMDID_PlaceHalfEllipse	Attributes field contains the icon's label or number.
"NONE"	Label field is "NONE."
""	Unused auxiliary information.

```
{{ 4*XC,2*YC,8*XC,4*YC}, Generic, GENERICID_DrawIcon, ON, 118, "dimtool.ma",""},
```

Figure 22. Example item instance for special icon generic item.

that defines the special icon. In Figure 22, the Mentor will display the icon whose number is 118 and is defined in the dimtool.ma (dimensioning tools) MDL file.

The Mentor looks for the MDL files (containing the special icons) in a directory that is defined by the MM_MAS MicroStation environment variable. When the Mentor is installed, it sets the MM_MAS variable to the "(ustation)\mdlsys\asneeded" directory. This is the directory where the icons not defined by MicroStation are located.

The generic item resource is similar to other resources used in MDL. Figure 23 shows the GENERICID_DrawIcon item resource for all icons, and Table 13 lists descriptions of the icon generic item fileds. The most important field for the Mentor is the hook function field, which calls the hook function with the ID HOOKITEMID_Generic_Icon. The hook function is what actually draws the icon into the Mentor card. It retrieves the icon ID number from the item instance.

Cell Generic Item. The cell generic item places pictures in the form of cells on the Mentor cards. Cells are frequently used complex objects in MicroStation drawings. Cells are stored in cell libraries and may be used repeatedly to speed the drawing process. Anyone familiar with basic MicroStation should understand how to create, store, and use cells. Placing cells on Mentor cards allows the developer to display pictures created with drawing tools available in basic MicroStation.

```
DItem  GenericRsc GENERICID  DrawIcon =
{
NOHELP, MHELP, HOOKITEMID  Generic  Icon, 0
 };
```

Figure 23. Example item for special icon generic item.

Table 13. Description of the icon generic item.

Field	Description
DItem GenericRsc	Describes type of resource. This is a generic resource.
GENERICID DrawIcon	Unique ID of resource.
NOHELP	No help for this item.
MHELP	If there were help, it would be BSI CAD help.
HOOKITEMID Generic Icon	ID of hook function that is linked to this item.
0	No argument.

```
{{10*XC,9*YC,60*XC,6*YC},Generic, GENERICID  DrawCell,ON,0,"4INTTP",""},
```

Figure 24. Example item instance for cell generic item.

Table 14. Description of the cell generic item instance.

Field	Description
{10*XC,9*YC,60*XC,6*YC}	Defines the location and size of the cell to be placed in card.
Generic	Defines the item type.
GENERICID DrawCell	ID of Item resource.
ON	Enables the item.
0	Attributes field is unused and should be set to zero.
"4INTTP"	Defines which cell to place on card.
""	Unused field.

Figure 24 shows the item instance for the cell generic item, and Table 14 lists descriptions of cell generic item fields. Again, it is similar to other item instances. One difference is that all the item instances that show cells use the same item resource. Therefore, there is only one item resource and many item instances for the cell generic item.

```
DItem GenericRsc GENERICID DrawCell =
{
NOHELP,MHELP,HOOKITEMID GenericCellDraw,NOARG
};
```

Figure 25. Cell generic item resource.

Table 15. Description of cell generic item resource.

Field	Description
DItem GenericRsc	Describes resource type. This is generic resource.
GENERICID DrawCell	ID of resource.
NOHELP	No help for this item.
MHELP	If there was help, it would be MicroStation help.
HOOKITEMID GenericCellDraw	ID of hook function that is linked to this item.
NOARG	No argument passed to hook function.

The cell generic item resource (Figure 25 and Table 15) only defines the hook function that draws the cell in the card. Where the icon generic item resource defines both the hook function and the icon to draw, cell generic item resource only defines the hook function. The item instance passes the variable to the hook function telling which cell to draw. Since the item resource does not define the variable and all the cell generic items use the same hook function, only one cell generic item resource is needed.

One should understand why the cell generic item has only one item resource with many item instances, whereas the icon generic item needs many different item resources with almost one instance for each item. The icon item resource passes the variable to the icon drawing hook function and the cell item instance passes the variable to the cell drawing hook function. The difference is in the type of variable passed. The cell passes a character string and the icon passes a number. These restrictions are caused by the dialog structures defined in MDL.

Commands

Commands are the part of the Mentor that open and close the Mentor cards. Commands are activated by dialog box items such as push buttons, pull down menus, and key-ins. Commands have two parts: the command table and the command handlers. The command table is located in a special resource source file (*.r) and the command handler is MDL source code located in a source code file (*.mc).

Command Tables

MicroStation stores all commands in command tables. It has a command parser that interprets user input and, if possible, assigns it to a command. The command table provides information necessary for the command parser to translate user information into MicroStation commands. For example, if the user keys in PLACE LINE, then the command parser will interpret the user input and submit the Place Line By Points command to MicroStation. The parser allows MicroStation to recognize abbreviated key-ins. The input should be a character string long enough to make it unique. The parser will then assign a command. If the key-in is not unique, the parser will issue the command that places the "ambiguous command" message in the MicroStation command window.

A command table is arranged in a hierarchical tree. Each level of the tree represents a key word of the input. For the Mentor, the first key word is "Mentor", so every Mentor command key-in must begin with "Mentor". Figure 26 shows some of the command table organization. Each table contains a series of words that leads to another sub-table. The root word for the Mentor is "Mentor". Each level in the command table is another word in the command. For example, follow the command table for the command key-in: MENTOR DESIGN CUBE. "Mentor" points to table CT Card and the "design" points to CT Design. (CT stands for command table.) The word "Cube" is the lowest level in this tree branch, so this command is executed. CT None stands for no command sub-table. The command parser allows the user to abbreviate key-ins as long as there is enough information to make them unique. An example abbreviation is: Mentor Des P E. Figure 26 shows why this is valid. This will execute the MENTOR DESIGN PLANE EXAMPLE command. "Mentor" points to the CT Card table and the "Des" points to CT Design. There are only two words in the CT Design table, "PLANE" and "CUBE". The "P" differentiates between the two options and points to CT Dplane. There is only one item in the CT Dplane table that begins with the letter "E", therefore the defined command is "Example."

The command table is a resource file. Figure 27 shows a portion of the Mentor's command table. Every command in the Mentor must be unique. The command must differ from any other Mentor, MicroStation, and MDL commands. To make the Mentor commands unique, they begin with "Mentor." Many of the commands in the Mentor are similar to MicroStation commands except for the "Mentor" root word.

CT Main is defined to be 1. Each table (sub-table) must have a unique ID number. Main contains only one item. Table 16 explains each field.

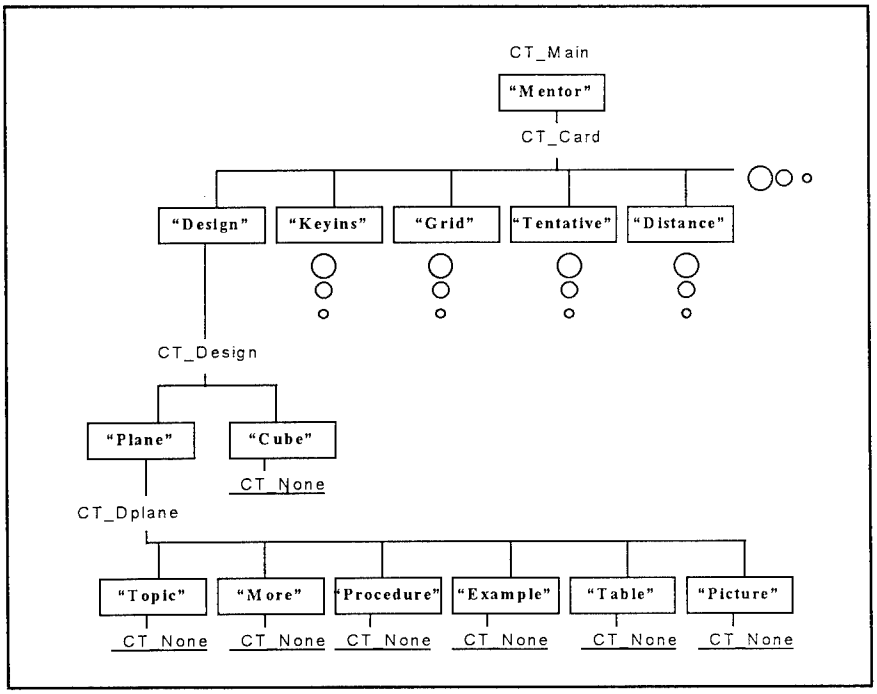


Figure 26. Organization of a command table.

The command table is compiled with the MDL rcomp compiler. It uses the -h extension of the compiler. This produces a header file used by the rest of the Mentor's source files. Figure 28 shows the portion of the header file produced using the Mentor's command table and the rcomp -h compiler. The numbers in the header files are used by the push buttons and pull-down menus to activate the commands. Recall that commands in the Mentor can be activated three ways:

1. Key-ins, which are interpreted by the command table
2. Push Buttons, which use the command numbers contained in the header file, such as the ones shown is Figure 28
3. Pull-downs, which use command numbers similar to push buttons.

Command Handlers

Command handlers are C-like functions that have the prefix of "cmdName". When the user invokes the Mentor command (say MEN FILE CREATION MORE), the code in the command handler with the same name (MEN FILE CREATION MORE in Figure 25) will execute. In the Mentor, command handlers often have the suffix "cmd Number". Directly following that is a label that represents a number (CMD MEN FILE CREATION MORE in Figure 29). MDL defines that label in the ptypetyp.h file. For each command in the Mentor, a corresponding command handler exists. For each command that displays a card, a command handler will open the card by executing a mdlDialog open statement. Figures 29 and 30 show a card's command handler.

```
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Main Command Table.    Defines "Mentor" key word              ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
Table CT_MAIN =
{
      /*    Subtable       command class  options    command word*/
      { 1, CT_CARD,        MANIPULATION,   REQ,      "MEN"    }
};
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Sub-table: Card. This is the Command table "Mentor"           ::
:: Points too.                                                   ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
Table CT_CARD =
{
      /*     Subtable       command class     options     command word */
      { 1, CT_INTRO4,        INHERIT,           DEF,       "LESSON4"      },
      { 2, CT_DESIGN,        INHERIT,           NONE,      "DESIGN"       },
      { 3, CT_KEYINS,        INHERIT,           NONE,      "KEYINS"       },
      { 4, CT_GRID,          INHERIT,           NONE,      "GRID"         },
      { 5, CT_TENTATIVE,     INHERIT,           NONE,      "TENTATIVE"    },
      { 6, CT_DISTANCE,      INHERIT,           NONE,      "DISTANCE"     },
      { 7, CT_NONE,          INHERIT,           REQ,       "RETURN"       },
      { 8, CT_NONE,          INHERIT,           REQ,       "TRACK "       },
      { 9, CT_NONE,          INHERIT,           REQ,       "NULL"         },
                             •                  •          •
                             •                  •          •
                             •                  •          •
};
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Sub-table:Design. This is the Command table "design"          ::
:: Points too.                                                   ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
Table CT_DESIGN =
{
      /*     Subtable       command class      options    command word */
      { 1, CT_DPLANE,        INHERIT,            DEF,       "PLANE"       },
      { 2, CT_NONE,          INHERIT,           NONE,       "CUBE"        },
};

/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Sub-table: Dplane. This is the Command table "Plane"          ::
:: Points too.                                                   ::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
Table CT_DPLANE =
{
      /*     Subtable       command class     options     command word */
      { 1, CT_NONE,          INHERIT,           DEF,       "TOPIC"        },
      { 2, CT_NONE,          INHERIT,          NONE,       "MORE"         },
      { 3, CT_NONE,          INHERIT,          NONE,       "PROCEDURE"    },
      { 4, CT_NONE,          INHERIT,          NONE,       "EXAMPLE"      },
      { 5, CT_NONE,          INHERIT,          NONE,       "TABLE"        },
      { 6, CT_NONE,          INHERIT,          NONE,       "PICTURE"      },
};
```

Figure 27. Partial Mentor command table.

Advanced MDL in the Mentor.

The majority of the Mentor uses standard items mentioned above, but some of the finer details require more complicated MDL features. With standard items, MicroStation

Table 16. Description of command table.

Field	Description
Table	This is a command table
CT MAIN	This is the ID number of the table.
Table's Commands	
1	This is the first and only item in this command table
CT CARD	If this keyword for this item is matched, search the sub-table with the ID CT CARD
MANIPULATION	This defines the command class of this item.
REQ	This option requires at least one more word to be a valid command.
MEN	This defines a keyword of this table command. Since there is only one item this is the only keyword of this table.

```
#define CMD_MEN                                0x01000000 /* MANIPULATION*/
#define CMD_MEN_DESIGN                         0x01020000 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE                   0x01020100 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE_TOPIC             0x01020110 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE_MORE              0x01020120 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE_PROCEDURE         0x01020130 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE_EXAMPLE           0x01020140 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE_TABLE             0x01020150 /* MANIPULATION */
#define CMD_MEN_DESIGN_PLANE_PICTURE           0x01020160 /* MANIPULATION */
#define CMD_MEN_DESIGN_CUBE                    0x01020200 /* MANIPULATION */
```

Figure 28. Portion of a header table produced from Mentor's command table.

defines the form and function, but with generic items, the hook function defines the form and function. To use the generic items described above, one need not write the hook functions because they already exist. Hook functions are very flexible and add a powerful dimension to MDL.

Hook Functions

Hook functions (sometimes called hooks) are associated with dialog boxes and dialog box items. They modify and/or amplify the behavior of dialog box items. Hence, there are two categories of hooks: dialog box hook functions and dialog item hook functions. Hooks are message handlers. Dialog hook functions receive messages about the dialog box as a whole, whereas the item hooks receive information about specific dialog items. MicroStation has a dialog manager that oversees all the events that occur in dialog boxes. When certain events occur, the dialog box manager sends messages to appropriate hook functions. These messages contain information that the hook function can either ignore, read, or modify.

```
cmdName MEN FILE CREATION MORE () cmdNumber CMD MEN FILE CREATION MORE
{
        Return command = Last command;
        if ((mdlDialog open (NULL, DIALOGID File Creat More)) == NULL)
        mdlOutput error ("Error MEN FILE CREATION MORE");
        Last command = CMD MEN FILE CREATION MORE;
}
```

Figure 29. A typical example of a command handler.

cmdName MEN FILE CREATION MORE () cmdNumber CMD MEN FILE CREATION MORE	Defines a function that will execute when the command MEN FILE CREATION MORE occurs
Return command = Last command;	Remembers the last command if the user decides to return to the previous card
if ((mdlDialog open (NULL, DIALOGID File Creat More)) == NULL)	Opens the corresponding Dialog Box
mdlOutput error ("Error MEN FILE CREATION MORE");	Report an error if MDL could not open that Dialog box
Last command = CMD MEN FILE CREATION MORE;	Sets MEN FILE CREATION MORE to be the last command

Figure 30. Description of command handler lines.

There are many different messages types sent from the dialog manager at different times. For example, when a dialog box is opened, the dialog manager sends certain messages to the dialog hooks. If there is no hook associated with the dialog box, then all messages are ignored. Then when the dialog box receives focus, the dialog box manager sends more messages. (Focus is when a dialog box or item is the active item that can receive input from the keyboard, mouse etc.) Also, when the a dialog box is closed, the dialog box manager sends different messages. For example, during the life of a dialog box (opening-closing), the dialog box manager sends these five dialog messages regardless of the events that occur:

DIALOG MESSAGE CREATE
DIALOG MESSAGE INIT
DIALOG MESSAGE BEFOREDESTROY
DIALOG MESSAGE HIDE
DIALOG MESSAGE DESTROY

The first two messages are sent at different times during the dialog box opening process. The last three messages are sent at the closing and removing of the dialog box. Other messages may be sent depending on the events that occur in the dialog box while it exists in memory. Some of these events include: dialog box gets focus, dialog box loses focus, dialog box resized, etc. See the MDL programmers guide under dialog hook function messages for messages sent by the dialog box manager.

Dialog item hooks are common and enhance the ability of standard dialog box items. Item hooks define the behavior of generic items. Item hooks are similar to dialog hooks because they handle many messages sent at different times. Different types of items receive different types of messages. Messages fall into three classes: general messages, input focusable messages, and generic messages. General messages are sent to all items. Input focusable messages are sent only to items that may gain or lose focus. Generic messages are sent only to generic items.

Hook functions receive very large structures that contain much information. Part of the structure is union, which is a packet of information that varies depending on message type. To handle different packets of information, hook functions are set up as switch statements. Depending on the message sent, hook functions will perform different tasks. Figure 31 shows the hook function that draws icons in the Mentor cards. This hook function only responds to the create and draw dialog item messages. When the function receives the create message, it modifies two variables. During the draw message, the function executes the MDL functions needed to draw the icons on the Mentor cards. The hook ignores all other item hook messages.

User Tracking

The Mentor system allows the user to step back through the visited cards in the current session. Three commands are available to do this: "Previous" to return to the previous card, "Back Track" to back track through many cards, and a history of the last five visited cards. Each of these commands is on the "Return" pull-down menu in the upper right corner of each Mentor card. The "Options" pull-down menu, found on the Mentor control box, also contains the history feature (Figures 32 and 33). In the Beta version of the Mentor, the command numbers of the visited Mentor cards were stored in a file called "usrtrk.men", which was used to track subjects' use of the Mentor.

Previous Card

When the user selects "Previous" from the "Return" pull-down menu while on a card (say A), the previous card (say B) will be displayed. If the user immediately selects "Previous" again, card A will appear because it was the card visited before card B. "Previous" will continue to flip back and forth between cards A and B until the user selects a different card.

The code that controls the previous card command is fairly simple. When a Mentor card is displayed, the command handler for that card remembers the last command number. When the "Previous" command is executed, the "Previous" command handler recalls the last command number and displays the corresponding card.

```
Public void Draw_Icon_ItemHook(DialogItemMessage *dimP)
        {
        dimP->msgUnderstood = TRUE;
        switch (dimP->messageType)
                {
                case DITEM_MESSAGE_CREATE:
                    {
                    dimP->dialogItemP->attributes.acceptsKeystrokes = FALSE;
                    dimP->dialogItemP->attributes.mouseSensitive = FALSE;
                    break;
                    }
                case DITEM_MESSAGE_DRAW:
                      {
O               O                       O               O
O               O                       O               O
O               O                       O               O
        /*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        ::      Execute MDL functions to obtain resources and draw icons in ::
        ::      Mentor card For simplicity, some code has been excluded    ::
        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
O               O                       O               O
O               O                       O               O
O               O                       O               O
                      break;
                      }
                default:
                      {
                      dimP->msgUnderstood = FALSE;
                      break;
                      }
                }
        }
```

Figure 31. Example dialog item hook function.

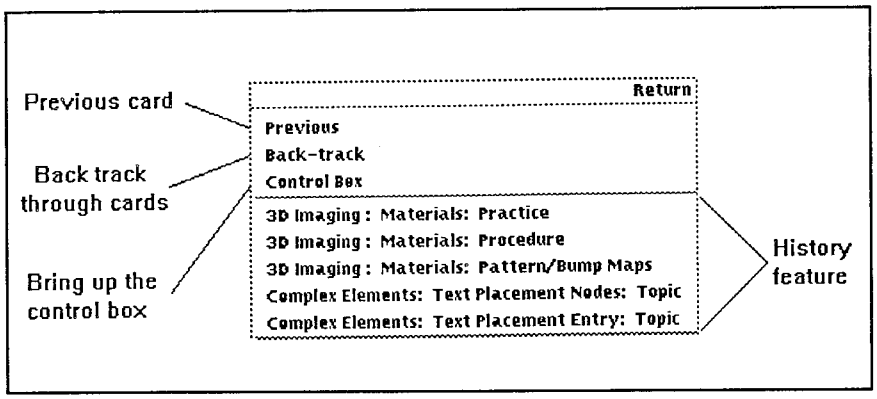


Figure 32. User tracking on the "Return" pull-down menu.

Back Tracking

The back track command on the "Return" pull-down menu will step the user backwards through the last 50 cards visited since the Mentor was loaded. As the user continues to select the back track command, each card that the user visited is displayed from the most to least recent. The history feature adds the cards to its list.

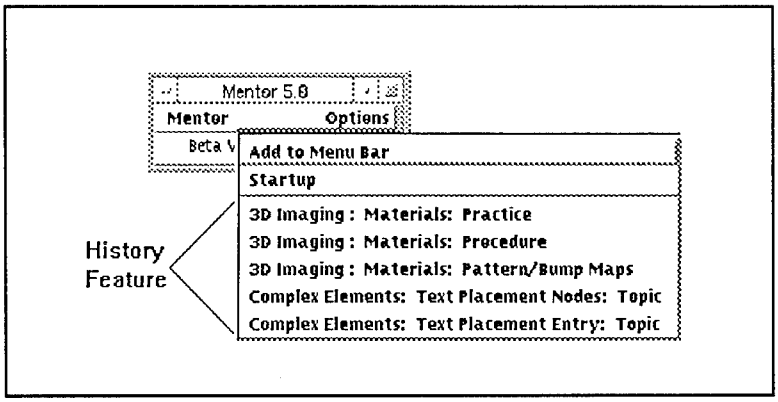


Figure 33. History on the "Options" pull-down menu.

The Mentor uses a pre-defined MDL function that sets up a filter that calls the user tracking function each time the user initiates a Mentor command. This line is:

```
MdlInput setFunction(INPUT COMMAND FILTER, write user track);
```

The write user track is the command filter hook for the Mentor. Any Mentor command the user invokes with a pull-down menu, key in, or button will trigger the execution of this user tracking function. Information about the initiated Mentor command, such as the command number, is sent to the user tracking function. The hook first checks this command number to make sure that it opens a card and is not a special command (like a movie display or tracking command). Commands that open Mentor cards are placed on the command stack, while special commands are not. The stack is similar to a stack of books. The most recently invoked command numbers are placed on top of the stack. This forces the command number in the top position to the second position. The command number in the second position is moved to the third position, and so on.

The MEN TRACK command handler function executes when the user issues a back track command. This function first assures that the stack has enough command numbers so that it can back track. If not, "No More Previous Cards" appears in the error message location. Otherwise, the function removes the top card from the stack and that card becomes the current card.

History

The history feature on the "Return" and "Options" pull-down menu displays the titles of the last five visited cards and allows the user to re-visit them. To implement the history list, the Mentor creates another stack (title stack) that contains the titles of the last five visited cards for display purposes. However, it uses the command number stack (discussed above) to issue the commands to display a card from the list. The

Dialog box hook maintains the title stack by putting the title of the card (that called the hook) on top of the title stack. The title stack is a stack of five titles, each having a name with a maximum of 100 characters.

When the user pulls down either "Return" or "Options," the pull-down menu calls a hook to display the correct card titles. Each hook gets information about the dialog box that contains the menu, the menu bar that contains the pull-down menu, and finally the menu with the history feature. Using the menu information and the item IDs associated with the history list, each hook changes the text of the history list in the appropriate pull-down menu. The new text in the history list will be the cards' titles stored in the title stack. The previous card's title will be the title at the top of the title stack. The second previous card's title will be the next title down on the title stack, and so on.

User Tracking File

The Beta version of the Mentor system stored the sequence of command numbers of visited cards into a file called "usrtrk.men." It also contained an opening dialog box that asked for the user's name. When the user opened a card, the write user track hook would invoke and write a line in the "usrtrk.men" file. The line contained the command number of the card opened, the user's name, and the current date (Table 17).

Table 17. Example of USRTRK.MEN file.

Command Numbers	User Name	Date
18481408	Jack	05 05 94
18481664	Jack	05 05 94
22151168	Jack	05 05 94
22216704	Sue H	05 05 94
17432832	Sue H	05 05 94
23068672	Jack	05 06 94
17235968	Sue H	05 09 94
17498368	Sue H	05 09 94
17563920	Sue H	05 09 94
17629456	Sue H	05 09 94

Miscellaneous Features

In addition to standard buttons that link cards to other cards, the Mentor has special buttons on certain cards. These special buttons display external files such as the included movie and design files. Three additional features will add the Mentor pull-down menu to the MicroStation command window, unload the Mentor, and display the Mentor control box.

Movie

The Mentor system includes a MicroStation movie file named MOVIE.FLI. The movie is a series of slightly different images that display rapidly, giving the appearance of motion. The card "3D Concepts : Perspective Projection : More Information" contains the special buttons that activate and deactivate the movie. The "SHOW MOVIE" button will open a dialog box and display the movie in that box. The "HIDE MOVIE" button will remove the movie dialog box.

When the user presses the "SHOW MOVIE" button, the command handler executes code that displays the movie. The name of the command handler function is MEN MOVIE. The function first finds the file MOVIE.FLI in the Mentor directory. Then it loads a MicroStation movie application by simulating the key in "MOVIE LOAD (Mentor dir) \MOVIE.FLI." The MicroStation movie application is an MDL program that opens a dialog box to display movies. When the movie application loads, its default playback setting is "loop sequence." Consequently the movie will keep repeating. The Mentor internally changes this setting so that the movie plays only once, giving the user the ability to repeat the movie if desired. To do this, the Mentor automatically sends a key in to open the movie settings dialog box. Then it sends a series of key strokes that disables the Loop Sequence setting and closes the settings dialog box. Finally, the Mentor sends the key in "MOVIE PLAY" to play a single sequence of the movie.

The user can hit the "SHOW MOVIE" button again while the movie dialog box is open, and the movie will play again. In this case, the button only sends the key in "MOVIE PLAY" because all the settings were changed the first time the "SHOW MOVIE" button was pressed.

The "HIDE MOVIE" button on the same card simply executes the MEN NOMOVIE command handler to unload the movie application from memory. This will close the movie dialog box. The next time the user hits the "SHOW MOVIE" button, the Mentor must repeat all the above steps to display the movie and to change the settings.

Design File

On the "Managing the Environment: View Control: More Information" card, the button "Load WINDOWS.DGN" will open a Mentor design file. The current design file will close and the Mentor drawing file will appear as a set of cascading windows of particular viewpoints. When the user hits the "Load WINDOWS.DGN" button, the command handler executes the MEN WINDOWS function. This function first copies the file WINDOWS.DGN from the Mentor directory into the default MicroStation design file directory. Then the Mentor opens the WINDOWS.DGN copy file in the default MicroStation design file directory. The original is kept in the Mentor directory as the master in case the user changes the opened copy. If this opened file is changed, the original remains untouched to replace the edited copy the next time the "Load ..." button is pressed.

Add the menu

Under the "Options" pull-down menu on the Mentor control box is "Add to Menu Bar." This choice will add the main pull-down menu of the Mentor to MicroStation's command window. When the user does this, the pull-down menu on the command window appears the same as the one on the Mentor's control box.

When the user chooses "Add to Menu Bar," the command handler executes the MEN MENUBAR function. This function adds the Mentor pull-down menu to the command window, and then dims this choice so that the user cannot add it again. The command handler retrieves information about the MicroStation command window and the its menu bar. Using this data, it adds the Mentor pull-down menu to the command window's menu bar. Data is retrieved about the "Options" menu, and the "Add to Menu Bar" choice is disabled.

Unloading

Choosing the "Unload Mentor" selection on the Mentor pull-down menu removes the Mentor system from memory. When the user unloads the Mentor, the command handler executes the MEN UNLOAD function. This function simply issues an unload statement.

However, when the Mentor first starts, it sets the standard C function, ptypeUnload, to execute when the Mentor unloads. The following line in the Mentor sets the function ptypeUnload as the unload function:

```
mdlSystem setFunction (SYSTEM UNLOAD PROGRAM, ptypeUnload);
```

The function ptypeUnload can be triggered anytime the Mentor unloads. This is not limited to the pull-down menu choice. The Mentor could unload when the user exits from MicroStation, another application unloads the Mentor, or MicroStation crashes. PtypeUnload deletes the Mentor pull-down menu from the command window (if the menu was added and the user was not exiting MicroStation). Lastly, this function closes the user tracking file (on the Beta version).

Control Box displaying

The control box can disappear behind other windows or dialog boxes or be accidentally closed. The "Control Box" option on the "Return" pull-down menu will bring the Mentor control box back. When the user chooses "Control Box," the command handler executes the MEN CONTROL function. This function simply opens the Mentor control dialog box.

Suggested Technical Readings

Dinh-Vu, Mach N., *Programming with MDL*, ISBN 0-934605-59-9 (High Mountain, 1991).

MicroStation MDL Programmer's Guide, Vol 5, Document no. DGA055010 (Intergraph, 1993).

Young, Kim H., *Mastering MicroStation MDL*, ISBN 1-878789-04-X (Chanimar Ace, 1991).

4 The Research Test

Research Plan

Rationale

To gather the type of information required for this study, data had to be collected while CAD professionals used the Mentor program. Such data collection can be done by human observation and protocol studies. In this case, since it was theorized that discovering usage patterns might require an extended period of time, it was not feasible to employ human observers. In addition, the tests could not be scheduled locally and would have required collecting data at remote sites. The option of collecting machine data has risks involved, but plans were made to use that technique even though some data might be lost.

The machine data was to be captured at the site and delivered by mail or electronically to USACERL where the analysis would be done. The paper data, records, surveys, etc., would also be faxed or mailed to USACERL. This plan was necessary for the convenience of the test sites because the test subjects were involved in work and would not be given special time to conduct this test. Since the theories being tested concerned the work environment, it was felt that such a plan would yield the most valid data as well as stay within the time and cost limitations of the test sites. An additional benefit from machine-collected data was that any patterns that were discovered would be valuable in associated machine learning studies being conducted at USACERL

It was hoped that one factor that would induce users to become test subjects was the need for CAD users to update to a new version of MicroStation and to learn the new 3D capabilities of the system. Plans were made to advertise widely for test subjects to get about 100 volunteers. In reality, the actual number of subjects completing the test was anticipated to be somewhat fewer. USACERL researchers visited three sites that expected to have at least 10 testers to explain the test to a point of contact and start the users. In the other cases, the test materials were mailed to individuals who asked to be in the test.*

* Appendix A includes a list of the volunteer participants in this part of the study.

The Survey

A pilot study conducted in January 1994 with seven architects and engineers at the U.S. Army Corps of Engineers Huntville Division, Huntsville, AL, site indicated that the system was easy to understand, and that the information was useful and worth keeping on all CAD computers. Users preferred questioning the system to using other sources of information including peers and phone support, but did not unanimously recommend that new users begin CAD with only the performance support system for instruction. Individuals reported using different types of information and examples. The open-ended questions along with a sample of validated survey items were used to design the final survey in the formal study.

The 30 survey items were designed to give objective feedback about features of the Mentor, reveal attitudes about the use of computer-aided instruction and the Mentor in particular, and to record what users thought they were doing as they used the Mentor. Eleven of the items had been used in previous studies of CAD users and could serve to validate the test instrument. About half of the questions were worded negatively for measurements of attitude or, if the item reported behavior, opposite to the expected behavior. This prevented users from patterning their responses, but the items had to be stated positively to arrive at a score for analysis purposes. The more positive in attitude or predicted in behavior, the higher the score. The test items appear in Appendix B.

Test Activities

Several Corps of Engineers' and Naval Facilities' designers volunteered to test the support system to learn to use a new CAD release. The test subjects varied from novices to those who had already attended class to learn the new update. Users came from several different disciplines including architects, civil engineers, structural engineers, mechanical engineers, technicians, and CAD operators. The subjects were initially assisted to start the Mentor program, but were intentionally left to develop their own style of using the program.

The users filled out a personal record that asked their past experience and their specific discipline. The test program automatically loaded with the CAD software and users were required to record their names each time they entered the system. Subjects were asked to use the system for 20 work days and to keep a brief log for each of those days. They were specifically directed to make a distinction between work-related tasks and pure learning tasks. At the end of the test period, subjects were asked to answer

a 30-item survey about their use of the system, their attitudes, learning preferences, and responses to the Mentor.*

The computer saved the user's name, the date, and a numeric screen identifier each time the user visited a card. The data was later read by a graphic charting program to plot the pattern, or user path, by day, for each user over the 20-day period. The data set was comprised of the personal record, the daily log for 20 days, the user path for the same 20-day period, and the final survey results.

The data analysis includes correlation between user tasks, background, discipline, and learning preferences, survey responses, and features of the user path. Those features will include patterns as well as usage frequencies over the 20-day period.

Research Questions

The questions that the analysis of the collected data was planned to address were:

1. To what extent do the users of performance support systems cover the complete subject content?
2. Are differences in tasks, backgrounds, or learning preferences reflected in differences in user paths?
3. Do the user paths show a characteristic change in pattern or frequency over time?
4. Is electronic performance support effective for new and/or experienced users of CAD?
5. Are the different types of information available in the Mentor judged desirable by the users?
6. Does the flexibility of the Mentor encourage different styles of use?

Findings and Analysis

Forty two test subjects returned data to CERL for analysis. Twenty nine of the data sets were complete, though there was apparently much misunderstanding about the daily log; that part was not useful for meaningful analysis. The only test subjects that responsibly returned data were the ones personally visited, and there were problems even with some of those. Individuals who attempted to test the program reported lack of time, loss of forms, inability to run MicroStation in a compatible version, and many related reasons. Completely unsupervised test subjects did not seem to understand

* Appendix B includes a copy of the test survey.

the rational for the test and were not motivated to be a part of it. Even individuals familiar with the researchers found excuses for not returning the data. Most of the reported analysis was drawn from the 29 complete and some partial data sets.

Demographics

Thirty two subjects were male and 10 were female. The average age of the subjects was 36, ranging from 22 to 53. Ten were architects, seven were civil engineers, four were electrical engineers, three were mechanical engineers, one was a hydrologist, 10 were technicians, and the remaining four did not identify their occupation. Twenty-four were college graduates while the rest were students or had not completed post-secondary education. All but four reported more than a year of computer experience and 13 had completed a course on the MicroStation update that the Mentor supported. Over half of the test subjects were experienced in earlier versions of MicroStation or another CAD system. On the whole, the make-up of the sample group was typical of most Corps of Engineers' offices.

Analysis by Extent of Use

Test subjects were rated by the size of the file that contained their user path; it was found that the group divided into two balanced groups in the completed data set. Fifteen users had looked at fewer than 100 Mentor cards while fourteen had studied 150 to over 300 cards. The average age of the extensive users was 36 and the other group averaged 34. Those experienced and trained with MicroStation were as likely to use the Mentor extensively as those not experienced with MicroStation. Women and men were equally represented in both groups. The only demographic variation between the two groups was the fact that the extensive user group contained six architects and the other group had three architects. The user paths for the architects indicated that they explored the architectural 3D part of the Mentor more than other users. There was a significant difference in the scores on the survey items by the two groups of users (t=2.82, Critical t=2.04 for a two-tailed test).

The survey question set was then divided into four groups of questions according to their general theme:

1. Warm fuzzy feeling about Mentor and MicroStation.
   (items 1, 2, 7, 9, 11, 13, 22, 27)
2. Positive about potential of independent computer aided learning.
   (items 6, 3, 4, 17, 21, 24)
3. Self-report about individual patterns of use.
   (items 5, 8, 15, 20, 23, 28, 18)

4. Positive about specific features of Mentor.
   (items 10, 12, 14, 16, 19, 25, 26, 29, 30)

Analysis of each item set by the two groups of users revealed that there was no difference between the heavy and the light users in their feelings about Mentor and MicroStation, their expectations about potential learning, or the way they thought they used the system. The significance came from their evaluation of specific Mentor features (t=3.58, Critical t=2.05 for a two-tailed test).

Survey Responses

Survey responses were ranked according to scores of users.

The following list (questions stated in positive form) is in order from 89 to 75 percent by agreement:

2. I like MicroStation.
29. Graphic examples are necessary for learning 3D.
7. The Mentor can be used for several different purposes.
14. Trying the MicroStation commands in the practice exercises helped me learn.
16. I found the picture information useful.
23. I did not complete the instruction sequentially.
9. The Mentor would be good to start a new MicroStation user.
12. The use of color on the information cards was necessary.

Items scoring from 75 to 50 percent:

1. I feel generally competent with the MicroStation concepts covered in the Mentor.
15. The Mentor was not too complicated for me.
27. I understand MicroStation very well.
5. I used my own work project to experiment with the MicroStation commands.
26. I found the example information useful.
30. I found the practice exercises useful.
17. The menu organization was easy to follow.
8. I had no trouble following the directions in the Mentor.
10. I found the procedural information useful.
19. I think animation would make some ideas more clear.
28. I did most of my experimentation in my own work project rather than in the prepared drawing file.
13. The Mentor is handy to have as I am working on a project.
21. A teacher need not be present to help learners get started.

11. I think all CAD computers should have the Mentor available.
6. The Mentor helped me learn more quickly than I do from classrooms or textbooks.
22 I would rather refer to the Mentor than ask someone for help.

Below 50 percent agreement:

3. Having previous experiences in drawing and/or drafting is not necessary before attempting the Mentor.
24. Having previous computer experiences is unnecessary before using the MicroStation Mentor.
18. I looked at most parts of the Mentor.
4. The Mentor documentation is unnecessary.
20. I frequently used the Mentor to look up specific areas of interest.

The following item was left blank so often it could not be ranked:

25. The architectural section of the Mentor gave me a better understanding of 3D concepts.

When the test scores were ranked for the group that used the Mentor extensively, the results were somewhat different for some items. The following rank order was obtained from that group:

The following list (questions stated in positive form) is in order from 93 to 80 percent:

9. The Mentor would be good to start a new MicroStation user.
10. I found the procedural information useful.
14. Trying the MicroStation commands in the practice exercises helped me learn.
26. I found the example information useful.
29. Graphic examples are necessary for learning 3D.
30. I found the practice exercises useful.
7. The Mentor can be used for several different purposes.
16. I found the picture information useful.
1. I feel generally competent with the MicroStation concepts covered in the Mentor.
2. I like MicroStation.
27. I understand MicroStation very well.
5. I used my own work project to experiment with the MicroStation commands.
15. The Mentor was not too complicated for me.
12. The use of color on the information cards was necessary.

Items scoring from 80 to 50 percent:

11. I think all CAD computers should have the Mentor available.
6. The Mentor helped me learn more quickly than I do from classrooms or textbooks.
17. The menu organization was easy to follow.
21. A teacher need not be present to help learners get started.
28. I did most of my experimentation in my own work project rather than in the prepared drawing file.
19. I think animation would make some ideas more clear.
13. The Mentor is handy to have as I am working on a project.
22. I would rather refer to the Mentor than ask someone for help.
8. I had no trouble following the directions in the Mentor.
25. The architectural section of the Mentor gave me a better understanding of 3D concepts.
18. I looked at most parts of the Mentor.
23. I did not complete the instruction sequentially.
24. Having previous computer experiences is unnecessary before using the MicroStation Mentor.

Items scoring below 50 percent:

3. Having previous experiences in drawing and/or drafting is not necessary before attempting the Mentor.
20. I frequently used the Mentor to look up specific areas of interest.
4. The Mentor documentation is unnecessary.

Some particular items provided revealing information. Item 18, "I looked at most parts of the Mentor" was ranked no. 26 by the total group and no. 25 by the group that used the Mentor extensively. Other areas of high agreement between the two groups of users were that:

- the documentation was necessary
- they did not frequently look up specific problem areas
- previous computer and drafting experience were relatively important.

Both groups were highly positive about MicroStation and the possibility of using the Mentor for different purposes. Responses to the question about the helpfulness of the 3D architectural example showed that users who did look at this example were as likely to leave the question blank as those who did not. However, users who did not

look at it judged it not helpful while 100 percent of the users who looked at the example extensively rated it as helpful.

The extensive Mentor users were about 10 percentage points more positive about asking the Mentor rather than someone else for help, feeling competent with the concepts and understanding MicroStation very well, and using their own work projects rather than the prepared practice file to experiment with MicroStation commands.

Large differences between the two groups were reported on some items. Extensive Mentor users were about 20 percent more likely to agree that a teacher need not be present, that the Mentor helped them learn more quickly than from classrooms of textbooks, that they did not follow sequential order in their learning, and that the Mentor should be available on all CAD computers. Ninety two percent of the extensive users felt that the Mentor would be good to start a new MicroStation user, however the new users in the sample seemed be equally divided between extensive and light users.

User Paths

Several characteristics were identified and used to examine and classify each of the 32 user paths that were collected. Already discussed were: (1) the size of the file (the number of cards visited); (2) the spread over the 20 days to determine how many separate occasions they had visited the Mentor; (3) whether the user visited the guide chapter (the instructions for using the Mentor), and; (4) whether they had looked at the 3D drawing section. Also of interest were the following results:

- whether they had moved through the information in a sequential manner, using the "next card" button most of the time
- whether they had used the menus to skip to different places
- whether they browsed by looking at the different types of information within a topic in a nonsequential manner
- whether they used the "hot button and return" feature
- whether test subjects changed their patterns of use over the 20 day period (and how many, if they had).

Examples of some of these characteristics as they appear on the users' path graphs are shown in Figures 34 and 35. The patterns represent a line graph indicating the order of the cards visited by the user. Two thirds of the users used sequential paths through the information cards. About half of them occasionally used browse buttons to look at specific types of information. In all but one case, the users who employed both sequential and browsing methods of navigation had looked at the guide. Nine of the 32 users did not visit the guide section of the Mentor. As expected, the users who used

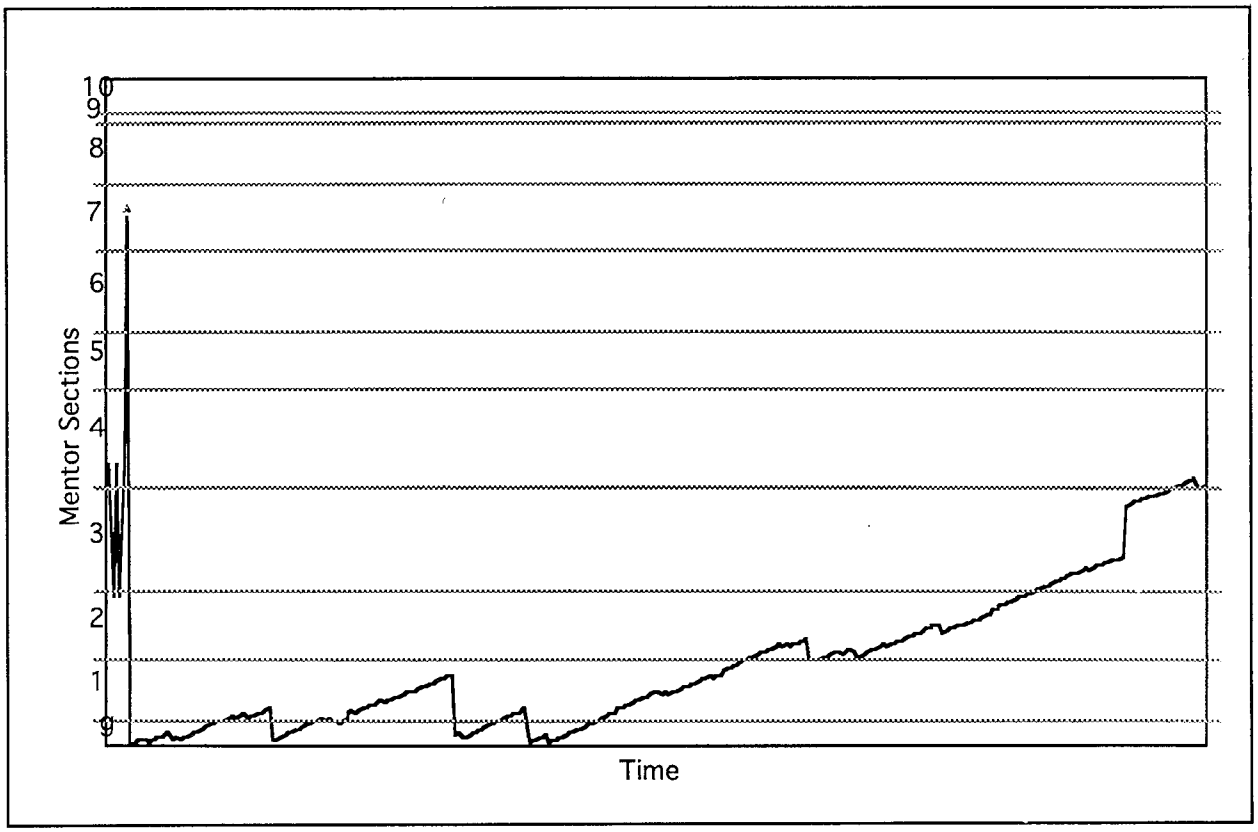


Figure 34. User path with a combination of sequential movement and menu skips after a brief initial hot button exploration.

the Mentor over a spread of several days were mainly the extensive users. However, one heavy user visited over 300 cards in one day.

Correlation between the different characteristics is shown in Table 18. The correlation between size and spread is fairly significant as the table shows: the extensive users were more likely to be sequential and unlikely to browse. In fact, browsing was not very evident in this sample even when they looked at the guide. There was some correlation between extensive users and the tendency to skip from topic card to topic card. The use of hot buttons was most prevalent among sequential users and those who had looked at the guide.

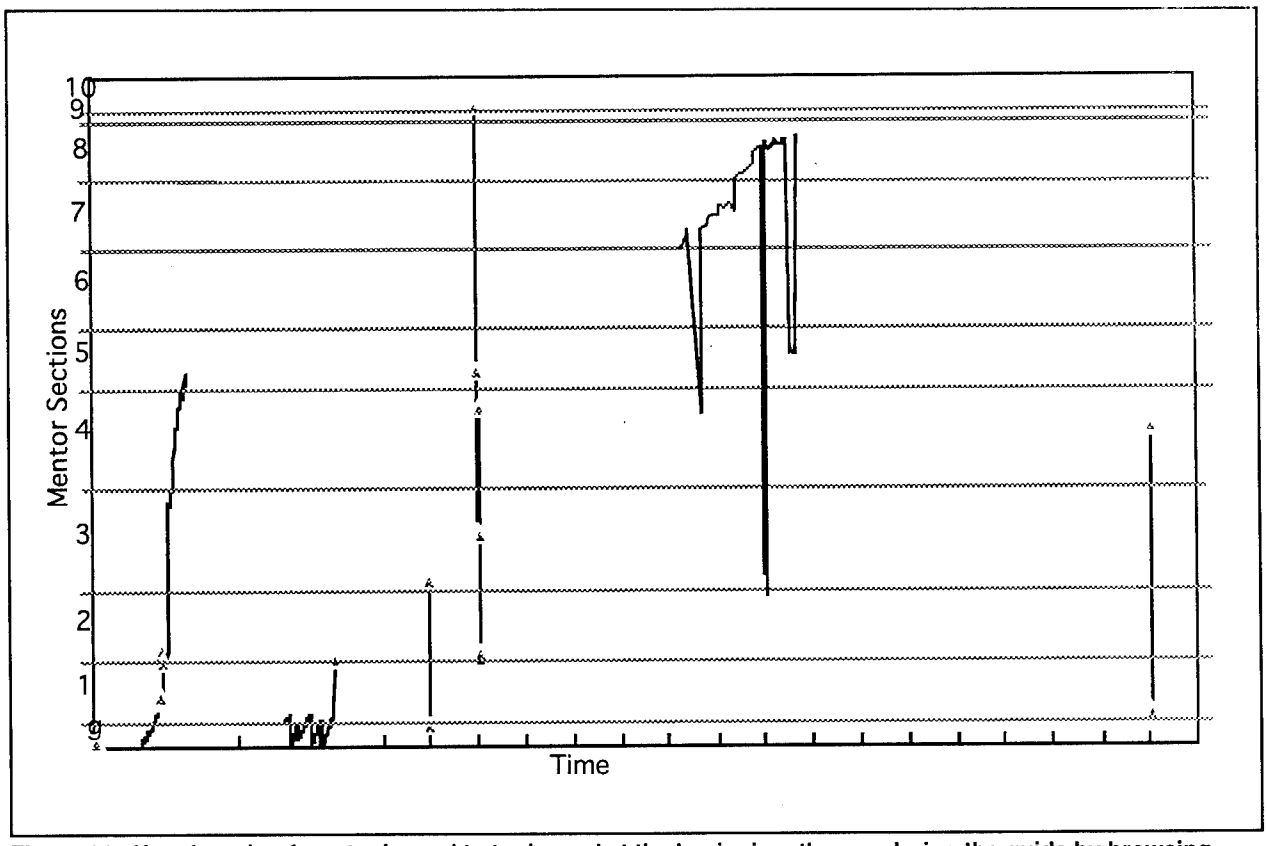


Figure 35. User hopping from topic card to topic card at the beginning, then exploring the guide by browsing, looking up some items of special interest, then studying the 3D section using hot button references.

Table 18. Correlation between user characteristics.

	size	guide	sequ	menu	browse	button	topic	spread	change
size	1								
guide	0.11	1							
sequential	0.38	0.43	1						
menu	0.19	-0.1	0	1					
browse	-0.4	0.13	-0.2	0.28	1				
button:return	0.12	0.27	0.24	0.13	0.24	1			
topic to topic	0.37	0.07	0.07	0.07	-0.07	0	1		
spread:time	0.65	0.03	0.02	0.28	-0.22	0.16	0.39	1	
change	0.08	0.03	-0.1	-0.2	0.075	-0.1	0.19	0.19	1

5 Conclusions and Recommendations

Conclusions

This study addressed the Mentor CAI system for use with Microstation CAD software, a CAD package widely used by the U.S. Army Corps of Engineers. The system was tested with users representing a range of disciplines and CAD expertise. Subjects used the system for a 20-day period, were requested to keep daily logs, and at the end of the test period, participated in a survey about their use of the system, their attitudes, learning preferences, and reactions to the Mentor.

About half of the subjects reported that they did not complete a significant part of the subject content. Even the group of extensive users reported that they had not covered much of the content. It was not anticipated that users would cover all the information in a performance support system, but it appears that users covered less than they might have if they had been given some way to determine exactly what they have done, an electronic "checklist" or "bookmark."

This study could not conclusively determine whether specific CAD work tasks being performed, the subjects' backgrounds, or individual learning preferences were most influential in the exploration paths that different users took. However, architects as a group did take one consistent path: they tended to look at the 3D example. This may have been influenced by the fact that the USACERL researcher who visited two of the test sites was an architect and may have made it a point to demonstrate this material.

The study showed no conclusive changes in usage patterns over the test period. It may be that 20 working days is not enough time for changes of this type to occur. Also, many users failed to keep a daily log; future studies may benefit from a more reliable (perhaps automated) way to determine and record the tasks that users work on and how well they accomplish their objectives.

Most users took a sequential path through the information, apparently setting aside time and selecting topics to view from beginning to end. This dedicated use of the system may explain the relatively low measured use of the Mentor system. Users may not feel comfortable setting aside blocks of time in the working day for tutorial study;

they apparently were not acquainted with the concept of "just in time" learning. This may be explained by the fact that most formal educational experience is conducted in sequential order; change from that anticipated pattern can be difficult.

Moreover, sequentially "plodding through" tutorial information may not be the most productive or efficient use of learning time. Individual users may not need, or may not be ready to learn particular bits of information until an actual need for that information arises. In such cases, simply browsing, or searching for information through an index may be a better approach. Also, when given the choice, individuals may vary the sequence in which information is presented and learned. This contrasts with the traditional view that information should be ordered in a static sequence or hierarchy. Many test subjects did in fact use a variety of approaches; although results of the survey showed that they were aware of that potential, most did not fully use the flexibility that was designed into the Mentor.

The survey showed the electronic performance support system to be a popular learning tool with the subjects in this study, particularly with those who used it extensively. Ninety-three percent of the heavy users felt that it would be a good program for new CAD users and 80 percent thought the Mentor should be made available on all CAD computers. The user survey also indicated that particular features designed into the Mentor system were judged favorably by users as a whole (more than 50 percent agreement in all cases but one), and by extensive users in isolation (more than 50 percent agreement in all cases, and more than 83 percent agreement in all cases but two). Extensive users, who are presumably better able to judge based on their experience with the different information types, reacted more favorably to Mentor than their less experienced counterparts. Future studies may benefit from taking more information (i.e., recording the user track) to determine whether and how much to weight specific survey questions, to give more credence to more qualified responses.

Recommendations

It is recommended that the capability to map information space be incorporated into performance support systems. Users need to be aware of what they have covered. As a result of subject coverage problems revealed in this study, a marking system has been added to the Mentor menus to identify the areas already covered. Furthermore, a current project will allow electronic monitoring of a drawing being produced while the user is doing project work, to determine not only what task is being done, and also to determine the quality of the work. It is also recommended that performance support systems include this capability to offer help when users may not recognize that they need it.

It is recommended that particular types of information aimed at special disciplines and learning styles be continued and enhanced. On the survey, test subjects indicated that they would like to see animation added to the graphic capabilities. Multi-media approaches add to concept building and motivation. Discipline-specific examples should be designed by professionals as was done in the current version of the Mentor.

Most importantly, it is recommended that new ways be derived to prepare users to distinguish the most important information from the information excess that they encounter in their daily work. No longer is it possible to learn sequentially everything that will be needed for a job. People must learn to assimilate what they require today and build on it tomorrow with something new. It is vital for professionals to overcome the segmented, stair-stepped, sanitized view of learning if they are to use electronic performance support systems to their full potential as effective learning and information tools.

Educational and Scientific Implications

It is hoped that this study may provide a greater depth of understanding of the behaviors and needs of users of performance support systems. This study was intended to lead to more intelligent CAI systems that might adapt to users' training requirements as they learn complex automated systems, thereby increasing the variety of learning choices.

The strength of hypertext electronic performance support at least partly stems from computer-based powers that allow users to follow any desired idea path without obstacles. However, such systems can be confusing; within a few associative jumps, hypertext can lead the user far from the current topic, consequently losing the context of the information search and failing to meet the user's needs. These difficulties can become compounded in large and complex systems such as CAD and engineering systems where resources can be overwhelmingly large, complex in purpose, and difficult to use. Future performance support systems must help manage and structure instruction so that people can receive the right amount of assistance at the right time. Naisbitt and Aburdene (1990) state the case for special management capabilities to assist people with the amount of information they face, "Without a structure, a frame of reference, the vast amount of data that comes your way each day will probably whiz right by you."

References

Alessi, S.M., "Fidelity in the Design of Instructional Simulations," *Journal of Computer-Based Instruction*, Vol 15, no. 2 (Spring 1988), pp 40-47.

Basalla, G., *The Evolution of Technology* (Cambridge University, London, 1988).

Brown, J.S., A. Collins, and P. Duguid, "Situated Cognition and the Culture of Learning," *Educational Researcher*, Vol 18, no. 1 (1989), pp 32-42.

Card, S.K., T.P. Moran, and A. Newell, A. *The Psychology of Human-Computer Interaction* (Lawrence Erlbaum Associates Publications, Hinsdale, NJ, 1983).

Fishman, A.M., and M.A. Silver, "The Psychology of Everyday CBT," *CBT Directions* (August 1990).

Gery, Gloria, *Electronic Performance Support Systems* (Weingarten Publications, Inc., Boston, MA, 1991).

Markle, S.M., *Designs for Instructional Designers* (Stipes Publishing Co., Champaign, IL, 1978).

Modesitt, K.L, *Smart People and Smart Computers: Networked Adaptive Performance Support Systems* (Loral Western Development Laboratories, Hanover, MD, February 1994).

Naisbitt, J., and P. Aburdene, *Megatrends 2000* (William Morrow Company, Inc., New York, 1990).

Nelson, Theodor H., *Dream Machines* (The Distributors, South Bend, IN, 1974).

Papert, Seymour, *Mindstorms* (Basic Books, New York, 1980).

Practice Management Associates, Ltd., *PSMJ CADD Application and User Survey* (Practice Management Associates, Ltd., Boston, 1994).

Reeves, T.C., "Pseudoscience in Computer-Based Instruction: The Case of Learner Control Research," *Journal of Computer-Based Instruction,* Vol 20, no. 2 (1993).

Shaw, Doris, and L.M. Golish, *Follow Up Studies of Embedded Instruction for CAD Systems*, Technical Report (TR) P-90/10/ADA222509 (U.S. Army Construction Engineering Research Laboratory [USACERL], May 1990).

Shaw, Doris, and L.M. Golish, *Intelligent Embedded Instruction for Computer-Aided Design (CAD) Systems* TR P-89/03/ADA201811 (USACERL, Champaign, IL, October 1988).

Skinner, B.F. "Teaching Machines," *Science*, 1958, Vol 128 (1958), pp 969-977.

Spiro, R.J., R.L. Coulson, P.J. Feltovich, and D.K. Anderson, *Cognitive Flexibility Theory: Advanced Knowledge Acquisition in Ill-Structured Domains*, TR no. 5 (Southern Illinois University School of Medicine, Springfield, IL, Conceptual Knowledge Research Project, 1988).

Spiro, R.J., R.L. Coulson, P.J. Feltovich, and D.K. Anderson, *Cognitive Flexibility Theory: Advanced Knowledge Acquisition in Ill-Structured Domains*, TR no. 5 (Southern Illinois University School of Medicine, Springfield, IL, Conceptual Knowledge Research Project, 1988).

Spiro, R.J., M.J. Jacobson, and J. Jehng, "Hypertext and Cognitive Flexibility: Theory and Technology for Learning in Complex Knowledge Domains," *30th International ADCIS Conference Proceedings* (November 1988).

Steinberg, E., "Cognition and Learner Control: A Literature Review, 1977-1988," *Journal of Computer-Based Instruction*, Vol 16, (1989), pp 117-121.

Steinberg, E. "Review of Student Control in Computer-Assisted Instruction," *Journal of Computer-Based Instruction*, Vol 3 (1977), pp 84-90.

Stoddard, M., *Learning Styles and Embedded Training: A Case Study,* Department of Energy (DOE) Report No. ED-274 305 (DOE, Los Alamos National Laboratory, 1985).

Tripp, S., *War of the Worlds, SIGTAR Newsletter* (Association for Computer Based Instructional Systems, Columbus, OH, 1992).

Appendix A: Volunteer Test Participants

Commander
USACE/Fort Worth District
ATTN: CESWF-ED-D/John Dagley
PO Box 17300
Fort Worth TX 76102-0300
PH: 817/334-8726; FAX: 817-334-3411

Commander
USACE/Huntsville Division
ATTN: CEHND-CS-ARCH/Mary Dougherty
PO Box 1600
Huntsville AL 35807-4301
PH: 205/955-4165; FAX: 205/955-3269

Commander
USACE/Mobile District
ATTN: CESAM-EN-DA/Gladston Hall
109 St. Joseph St.
Mobile AL 36628-001
PH: 205/694-4090; FAX: 205/690-2424

Commander
USACE/New England Division
ATTN: CENED/Chiway Hsiung
424 Trapelo Road
Waltham MA 02254-9149
PH: 617/647-8545

Commander
USACE/Rock Island District
ATTN: John A. Kincaid
Clock Tower Building
Rock Island IL 61201
PH: 309/794-5492

2852 ABG/CES
ATTN: Betty Marchbanks
McClellan AFB CA 95652
PH: 916/643-4875

Commander
USACE/New Orleans District
ATTN: CELMN-ED-DD/Jorge a. Romero
PO Box 60267
New Orleans, LA 70160-0267
PH: 504/862-2645

Vivian Sanches CODE 2421VS
Commanding Officer
Southwest NAVFACENGCOM
1200 Pacific Highway
San Diego CA 92132-5190
PH: 619/532-1168

Commander
USACE/Trans Atlantic Division
ATTN: CESAT-EC-TA/Jane Smith
PO BOX 2250
Winchester VA 22601
PH: 703/665-3698; FAX: 703/665-3621

Dennis Thornton CODE 04BDT
Commanding Officer
EFA Northwest
3505 NW Anderson Hill rd
Silverdale WA 98383-9130
PH: 206/476-8290 (X-385)

Donald Tsao CODE 09F4D
Commander
West NAVFACENGCOM
900 Commodore Dr
San Bruno CA 94066-2402
PH: 415/244-3060

Commander
USACE/Sacramento District
ATTN: CESPK-ED-A/Clare Van Dyke
1325 J St. CADD Section
Sacramento CA 95814-2922
PH: 916/557-7348

Naval Facilities Engr Command
ATTN: CODE DSO-1E/Louise McMonegal
200 Stovall St
Alexandria VA 22332-2300
PH: 703/325-0450; FAX: 703/325-4450

Commander
US Army Corps of Engineers
ATTN: CESWF-IM-IS/Martin Montes
PO Box 17300
Fort Worth TX 76102-0300
PH: 817/334-9956

Commander
USACE/PACIFIC OCEAN DIVISION
ATTN: Santiago Mor
Building 230
Fort Shafter HI 96858-5440
PH: 808/438-0790

Commander
USAEDE/CETAE-PM-MC
ATTN: Roy Parks
Unit 25727, APO AE 09242
PH: 011-49-69-151-5085

Commander
USACE/Portland District
ATTN: Norman E. Ragnone
319 S.W. Pine PO Box 2946
Portland OR 97208-2946
PH: 503/326-3427

Commander
USACE/Albuquerque District
ATTN: ED-T/Paul Rebarchik
517 Gold Ave. SW
Albuqueque NM 87103-1580
PH: 505/766-8289; FAX: 505/766-8733

Commander
USACE/Kansas City District
ATTN: Martha Walkup
Federal Bldg 601 East 12th St.
Kansas City MO 64106
PH: 816/426-5552

Commander
USACE/Wilmington District
ATTN: CESAW-EN-DG/Doug Wall
PO Box 1890
Wilmington NC 28402
PH: 910/251-4440

Mike Wang CODE 1632F
Commander
LANTNAVFACENGCOM
1510 Gilbert St
Norfolk, VA 23511-2699
PH: 804/322-4688

15 CES/CEF
ATTN: Len A. Waterman
75 H St.
Hickam AFB HI 96853-5233
PH: 808/449-6391

Heath Wells CODE 1812
Commanding Officer
ENGFLDACT Chesapeake
901 M St. SE
Washington DC 20374-5018
PH: 202/685-3281

Appendix B: Test Survey

NAME_____ MICROSTATION MENTOR

RESPOND TO THE FOLLOWING STATEMENTS BY CHECKING THE APPROPRIATE COLUMN. PLEASE DO NOT OMIT ANY ITEMS; PICK THE ONE THAT BEST APPLIES.

		STRONGLY AGREE	AGREE	DISAGREE	STRONGLY DISAGREE
1.	I feel generally competent with the MicroStation concepts covered in the Mentor.				
2.	I like MicroStation.				
3.	Having previous experiences in drawing and/or drafting is necessary before attempting the MicroSta tion Mentor.				
4.	The Mentor documentation is unnecessary.				
5.	I did not use my own work project to experiment with the MicroStation commands.				
6.	The Mentor helped me learn more quickly than I do from classrooms or textbooks.				
7.	The Mentor can be used for several different purposes.				
8.	I had trouble following the directions in the Mentor.				
9.	The Mentor would not be good to start a new MicroStation user.				
10.	I found the procedural information useful.				
11.	I think all CAD computers should have the Mentor available.				
12.	The use of color on the information cards was not important.				
13.	The Mentor is handy to have as I am working on a project.				
14.	Trying the MicroStation commands in the practice exercises helped me learn.				

NAME_____

		STRONGLY AGREE	AGREE	DISAGREE	STRONGLY DISAGREE
15.	The Mentor was too complicated for me.				
16.	I did not find the picture information useful.				
17.	The menu organization was easy to follow.				
18.	I didn't look at many parts of the Mentor.				
19.	I think animation would make some ideas more clear.				
20.	I frequently used the Mentor to look up specific areas of interest.				
21.	A teacher should be present to help learners.				
22	I would rather ask someone for help than refer to the Mentor.				
23.	I completed the instruction sequentially.				
24.	Having previous computer experiences is necessary before using the MicroStation Mentor.				
25.	The architectural section of the Mentor gave me a better understanding of 3D concepts.				
26.	I found the example information useful.				
27.	I don't understand MicroStation very well.				
28.	I did most of my experimentation in the prepared drawing file rather than in my own work project.				
29.	Graphic examples are not necessary for learning 3D.				
30.	I found the practice exercises useful.				

USACERL DISTRIBUTION

Chief of Engineers
ATTN: CEHEC-IM-LH (2)
ATTN: CEHEC-IM-LP (2)
ATTN: CECG
ATTN: CECC-P
ATTN: CECC-R
ATTN: CECW
ATTN: CECW-O
ATTN: CECW-P
ATTN: CECW-PR
ATTN: CEMP
ATTN: CEMP-ES (2)
ATTN: CEMP-C
ATTN: CEMP-M
ATTN: CEMP-R
ATTN: CERD-C
ATTN: CERD-ZA
ATTN: CERD-L
ATTN: CERD-M
ATTN: CERM
ATTN: DAEN-ZC
ATTN: DAIM-FDP

CECPW 22310-3862
ATTN: CECPW-E
ATTN: CECPW-FT
ATTN: CECPW-ZC

US Army Engr District
ATTN: Library (40)

US Army Engr Division
ATTN: Library (12)

US Army Europe
ATTN: AEAEN-EH 09014
ATTN: AEAEN-ODCS 09014
29th Area Support Group
ATTN: AERAS-FA 09054
100th Support Group
ATTN: AETT-EN-DPW 09114
222d Base Battalion
ATTN: AETV-BHR-E 09034
235th Base Support Battalion
ATTN: Unit 28614 Ansbach 09177
293d Base Support Battalion
ATTN: AEUSG-MA-AST-WO-E 09086
409th Support Battalion (Base)
ATTN: AETTG-DPW 09114
412th Base Support Battalion 09630
ATTN: Unit 31401
Frankfurt Base Support Battalion
ATTN: Unit 25727 09242
CMTC Hohenfels 09173
ATTN: AETTH-DPW
Mainz Germany 09185
ATTN: BSB-MZ-E
21st Support Command
ATTN: DPW (9)
US Army Berlin
ATTN: AEBA-EH 09235
ATTN: AEBA-EN 09235
SETAF
ATTN: AESE-EN-D 09613
ATTN: AESE-EN 09630
Supreme Allied Command
ATTN: ACSGEB 09703
ATTN: SHIHB/ENGR 09705

INSCOM
ATTN: IALOG-I 22060
ATTN: IAV-DPW 22186

USA TACOM 48397-5000
ATTN: AMSTA-XE

Defense Distribution Region East
ATTN: DDRE-WI 17070

HQ XVIII Airborne Corps 28307
ATTN: AFZA-DPW-EE

4th Infantry Div (MECH) 80913-5000
ATTN: AFZC-FE

US Army Materiel Command (AMC)
Alexandria, VA 22333-0001
ATTN: AMCEN-F
Installations: (19)

FORSCOM
Forts Gillem & McPherson 30330
ATTN: FCEN
Installations: (23)

6th Infantry Division (Light)
ATTN: APVR-DE 99505
ATTN: APVR-WF-DE 99703

TRADOC
Fort Monroe 23651
ATTN: ATBO-G
Installations: (20)

Fort Belvoir 22060
ATTN: CETEC-IM-T
ATTN: CETEC-ES 22315-3803
ATTN: Water Resources Support Ctr
ATTN: Australian Liaison Office

USA Natick RD&E Center 01760
ATTN: STRNC-DT
ATTN: DRDNA-F

US Army Materials Tech Lab
ATTN: SLCMT-DPW 02172

USARPAC 96858
ATTN: DPW
ATTN: APEN-A

SHAPE 09705
ATTN: Infrastructure Branch LANDA

Area Engineer, AEDC-Area Office
Arnold Air Force Station, TN 37389

HQ USEUCOM 09128
ATTN: ECJ4-LIE

AMMRC 02172
ATTN: DRXMR-AF
ATTN: DRXMR-WE

CEWES 39180
ATTN: Library

CECRL 03755
ATTN: Library

USA AMCOM
ATTN: Facilities Engr 21719
ATTN: AMSMC-EH 61299
ATTN: Facilities Engr (3) 85613

USAARMC 40121
ATTN: ATZIC-EHA

Military Traffic Mgmt Command
ATTN: MTEA-GB-EHP 07002
ATTN: MT-LOF 20315
ATTN: MTE-SU-FE 28461
ATTN: MTW-IE 94626

Fort Leonard Wood 65473
ATTN: ATSE-DAC-LB (3)
ATTN: ATZA-TE-SW
ATTN: ATSE-CFLO
ATTN: ATSE-DAC-FL

Military Dist of WASH
Fort McNair
ATTN: ANEN 20319

USA Engr Activity, Capital Area
ATTN: Library 22211

US Army ARDEC 07806
ATTN: SMCAR-ISE

Engr Societies Library
ATTN: Acquisitions 10017

Defense Nuclear Agency
ATTN: NADS 20305

Defense Logistics Agency
ATTN: DLA-WI 22304

Walter Reed Army Medical Ctr 20307

National Guard Bureau 20310
ATTN: NGB-ARI

US Military Academy 10996
ATTN: MAEN-A
ATTN: Facilities Engineer
ATTN: Geography & Envr Engrg

Naval Facilities Engr Command
ATTN: Facilities Engr Command (8)
ATTN: Division Offices (11)
ATTN: Public Works Center (8)
ATTN: Naval Constr Battalion Ctr 93043
ATTN: Naval Facilities Engr Service Center 93043-4328
ATTN: Code DSO-1E 22332-2300
ATTN: Code 2421VS 92132-5190
ATTN: Code 09F4D 94066-2402
ATTN: Code 1632F 23511-2699

8th US Army Korea
ATTN: DPW (12)

USA Japan (USARJ)
ATTN: APAJ-EN-ES 96343
ATTN: HONSHU 96343
ATTN: DPW-Okinawa 96376

416th Engineer Command 60623
ATTN: Gibson USAR Ctr

US Army HSC
Fort Sam Houston 78234
ATTN: HSLO-F
Fitzsimons Army Medical Ctr
ATTN: HSHG-DPW 80045

Tyndall AFB 32403
ATTN: HQAFCESA Program Ofc
ATTN: Engrg & Srvc Lab

USA TSARCOM 63120
ATTN: STSAS-F

American Public Works Assoc. 64104-1806

US Army Envr Hygiene Agency
ATTN: HSHB-ME 21010

US Gov't Printing Office 20401
ATTN: Rec Sec/Deposit Sec (2)

Nat'l Institute of Standards & Tech
ATTN: Library 20899

Defense Tech Info Center 22304
ATTN: DTIC-FAB (2)

15 CES/CEF 96853-5233

Air Force Academy 80920

McClellan AFB 95652
ATTN: 2852 ABG/CES

USAEDE
ATTN: CETAE-PM-MC 09242

Corps of Engineers
ATTN: 416 ENCOM 53403-0731
ATTN: CESWL-ED-DA 72201